Institut für Systemdynamik
Prof. Dr.-Ing. O. Sawodny

Universität Stuttgart

Bachelorarbeit

A Model-based Simultaneous Localization and Mapping Approach for Deformable Bodies

A Model-based Simultaneous Localization and Mapping Approach for Deformable Bodies

vorgelegt von
Jonathan Haag

Studiengang
Technische Kybernetik

Prüfer
Prof. Dr.-Ing. Oliver Sawodny
Prof. Dr.-Ing. Cristina Tarín

Betreuer
Johannes Schüle, M.Sc., Peter Somers, M.Sc.

Prüfungsdatum
29.03.2021

Institut für Systemdynamik, Universität Stuttgart, Waldburgstraße 17/19, 70563 Stuttgart

Kurzfassung

In der minimal-invasiven Chirurgie stehen typischerweise lediglich die Aufnahmen der verwendeten Endoskopiekamera zur Verfügung, um die präzise intraoperative Navigation zu ermöglichen. Ziel ist es, dem Operateur laufend verlässliche Informationen über die gegenwärtige Position der Kamera im Raum sowie in Hinblick auf das Zielgewebe zur Verfügung zu stellen. Ein bekanntes Lokalisierungsverfahren aus der Robotik ist der Simultaneous Localization and Mapping (SLAM)-Algorithmus, der gleichzeitig eine Karte der Umwelt erstellt und sich innerhalb dieser lokalisiert. Allerdings sind klassische Verfahren für starre Umwelten konzipiert und eignen sich deshalb nicht für die Operationsumgebung, in der es während des Eingriffs zu Verformungen kommen kann.

Diese Arbeit präsentiert das Konzept eines Modell-basierten SLAM-Algorithmus, welcher für deformierbare Umgebungen entwickelt ist. Dieser Ansatz prädiziert die Verformung in einer parallelen Simulation unter Verwendung eines Modells des realen Systems und nutzt diese Information, um weiterhin die Erfüllung der beiden zugrundeliegenden Aufgaben sicherzustellen. Der Fokus der Arbeit liegt auf der Etablierung des benötigten Frameworks zur Berücksichtigung der mittels Simulation vorhergesagten Deformation als Erweiterung des klassischen SLAM-Algorithmus. Darüber hinaus wird die Funktionsweise des Ansatzes in synthetischer Umgebungen unter Annahme eines idealen Modells und genauer Kenntnis der wirkenden Kräfte gezeigt. Diese Szenarien demonstrieren das Potential des Konzepts in Form von akkuratem Kameratracking und rauscharmer Kartierung. Diese Arbeit legt den Grundstein für Entwicklung weiterer Methoden zur Erweiterung des Modell-basierten SLAM-Algorithmus auf reale Systeme in zukünftigen Veröffentlichungen.

Abstract

In minimally-invasive surgeries typically only the recordings of an endoscopic camera are available when trying to develop methods that allow precise intraoperative navigation. The overall goal is to continuously provide reliable information, not only about the current position of the camera in space, but also regarding its orientation with respect to the target tissue. Simultaneous Localization and Mapping (SLAM) is a well known technique in the field of robotics that focuses on concurrently creating a map of the robots environment while locating itself within this map. However, classical approaches are structured in a way that they rely on their surroundings to be rigid, an assumption that is not fulfilled in the surgical environment.

This work presents the concept of a Model-based SLAM algorithm which is developed specifically for deformable settings. In this approach the deformation is predicted in a parallel simulation using a model of the real system. This information is then utilized to continue to perform the two main tasks. The focus of this work is on the establishment of the framework required to consider the predicted deformation as an extension to the classical SLAM algorithm. Additionally, the functioning of the proposed approach in synthetic environments is shown under the assumption of an ideal model and exact knowledge of the acting forces. Those scenarios demonstrate the potential of the concept in the form of accurate camera tracking and low-noise mapping. This work lays the foundation of future publications developing additional methods to extend the Model-based SLAM onto real systems.

Inhaltsverzeichnis

Abkürzungen und Formelzeichen

Abkürzungen

FEM	Finite Elemente Methode
ISYS	Institut für Systemdynamik
SLAM	Simultaneous Localization and Mapping
SOFA	Simulation Open Framework Architecture

Kapitel 1

Einleitung und Motivation

In den letzten Jahrzehnten wird der Einsatz minimal-invasiver Eingriffe in der Medizin immer häufiger. Der Wunsch, Operationstraumata zu minimieren und die gleichzeitige Gewährleistung einer erfolgreichen Operation, stellen Chirurgen und die Medizintechnik vor gleichermaßen große Herausforderungen. Insbesondere die Onkologie, die es sich zum Ziel macht, Tumoren vollständig zu entfernen, sieht sich mit einer Vielzahl von Schwierigkeiten konfrontiert. Um dem Operateur verlässliche Informationen über die Umgebung zukommen zu lassen, ist der Einsatz von Endoskopie-Kameras unabdingbar. Das Ziel ist es, das maligne Gewebe rückstandsfrei zu entfernen, dabei die Beschädigung des umliegenden, gesunden Gewebes aber zu minimieren. Um dies sicherzustellen, ist eine präzise Lokalisierung mithilfe der Bilddaten der endoskopischen Kamera von enormer Bedeutung - schließlich befindet sich die Gewebeunterscheidung im Bereich weniger Millimeter. Es müssen also Verfahren entwickelt und angewandt werden, die eben diese Lokalisierung auf effiziente Weise durchführen können, wobei zwei Fragestellungen im Zentrum stehen: Zunächst sind die Position und Orientierung der Kamera selbst, sowie anderer beteiligter Operationswerkzeuge, beispielsweise zusätzlicher Sensoren, von Interesse. Darüber hinaus gilt es, dem Chirurgen die intraoperative Navigation zu erleichtern, indem die Position des Zielgewebes, also des Tumors, bestimmt wird. Schließlich sind genau diese beiden Informationen entscheidend, wenn der befallene Bereich entfernt werden soll.

Die Schwierigkeit der Aufgabe ergibt sich aus ihrer Formulierung: Für die Lokalisierung sowohl der Kamera als auch die Kartierung der Umgebung steht nur das Kamerabild, aber keine sonstigen Sensordaten, zur Verfügung. Ähnliche Probleme lassen sich in der Robotik oder dem Autonomen Fahren wiederfinden. Dort existierende Algorithmen, die eben jene Lokalisierungs-Aufgabe verlässlich und genau lösen, sind aber nicht direkt übertragbar. Der Unterschied zum hier betrachteten Fall ergibt sich bei genauerer Betrachtung der Umgebung. Während die bestehenden Verfahren von einer starren Umwelt ausgehen und auf dieser Grundlage aufbauen, verletzt die Operationsumgebung diese Grundannahme. So werden im Fall des umherfahrenden Autos herausstechende Merkmale wie Häuserkanten, parkende Autos, Straßenschilder, Bordsteinkanten oder Bäume am Straßenrand verwendet, um sich zu orientieren. Im Gegensatz dazu dient in der Endoskopie die Oberflächenstruktur des behandelten Organs als Handlungshilfe. Betrachtet man zum Beispiel die Blase, so ist es jedoch offensichtlich, dass es durch Befüllen und Entleeren, also einer Veränderung des Volumens, zu einer Verformung des Gewebes kommt. Da eben dieser Prozess während der Operation bewusst eingesetzt wird, um bestimmte Stellen leichter zu erreichen, kommt es hier zu Problemen. Noch dazu lässt sich eine Deformation auch durch Krafteinwirkung mittels der Operationswerkzeuge beobachten, wenn etwa geschnitten oder mit einem Sensor auf das Organ gedrückt wird. Da sich durch die

Verformung auch die Position der Blutgefäße verändert, wird die Lokalisierung maßgeblich erschwert und klassische Ansätze versagen.

Diese Arbeit präsentiert das Konzept eines Modell-basierten Lokalisierungsalgorithmus für verformbare Systeme, der die Gewebe-Deformation durch Simulation berücksichtigt und dadurch eine präzise Navigation in der Endoskopie-Umgebung ermöglicht. Diese Arbeit versteht sich als *Proof of Concept* und zielt darauf ab, das entwickelte Konzept in einer synthetischen Umgebung unter Annahme eines idealen Modells zu etablieren und mit Simulationsergebnissen zu beweisen. Der Fokus liegt deshalb auf der Entwicklung eines Frameworks für das Konzept und der Herausarbeitung von Ansätzen, wie die prädizierte Verformung erfolgreich einbezogen werden kann.

Die restliche Arbeit ist wie folgt aufgebaut: Zunächst wird in Kapitel 2 ein klassischer Algorithmus für das Lokalsierungsproblem für starre Umwelten vorgestellt und erläutert. Dabei werden die konkreten Probleme, die sich bei einer Verformung der untersuchten Umwelt ergeben, herausgearbeitet. Anschließend befasst sich Kapitel 3 mit einem Modell-basierten Ansatz, der auch für deformierbare Umgebung anwendbar ist und präsentiert das zugehörige Framework, die einzelnen Komponenten sowie Simulationsergebnisse, die das gewählte Konzept unterstützen. Der gewählte Ansatz baut auf dem in Kapitel 2 vorgestellten auf. Schließlich werden in Kapitel 4 die Resultate zusammengefasst und nächste Schritte zur Erweiterung des Konzepts, vor allem in Hinblick auf seine Anwendung auf reale Systeme, diskutiert.

Kapitel 2

Klassischer Lokalisierungs-Algorithmus

Dieses Kapitel präsentiert eine Einführung in einen Lokalisierungsalgorithmus aus der klassischen Robotik, den Simultaneous Localization and Mapping (SLAM)-Algorithmus. Mit diesem Ansatz ist es möglich, für starre Umwelten aus Kamera-Bilddaten die dreidimensionale Umgebung zu rekonstruieren und die Kamera innerhalb dieser Umgebung zu lokalisieren. Erste Überlegungen hierzu lassen sich bereits in [CL85] und [SSC90] finden, [Dav03] präsentiert eine echtzeitfähige Lösung für den monokularen Fall. Einen Einstieg in die Thematik liefern die Veröffentlichungen [DB06], [BD06] und [Thr08], die sich mit der Formulierung der Aufgabenstellung und grundsätzlichen Lösungsmethoden befassen. Paper [FPRARM15] präsentiert eine Übersicht über die Historie und aktuelle Methoden der SLAM-Forschung, [Cad+16] ergänzt dies um zukünftige Schwerpunkte. Offensichtlich ist die grundsätzliche Aufgabenstellung ähnlich zu dem im Rahmen dieser Arbeit betrachteten Fall. Das Ziel dieses Abschnitts ist es, einen etablierten Ansatz zu erläutern und seinen Ablauf darzustellen. Hierzu muss zunächst die Frage geklärt werden, wie es überhaupt möglich ist, aus dem zweidimensionalen Kamerabild die dreidimensionale Umgebung zu rekonstruieren. Im Anschluss befasst sich das nächste Kapitel mit den Problemen, die sich bei unveränderter Anwendung auf das Operations-Szenario ergeben.

2.1 Grundidee

In vielen verschiedenen Anwendungen werden einzelne Kameras zur Navigation verwendet. Als Beispiele lassen sich hier planetarische Rover, autonome Fahrzeuge oder Roboter nennen. Da das zugrundeliegende Problem bereits seit mehreren Jahrzehnten untersucht wird, sind für die klassischen Anwendungsfälle bereits viele Lösungsmethoden vorgestellt und diskutiert worden. Eine Gemeinsamkeit all dieser Ansätze ist, dass hervorstechende Merkmale im Bild als solche erkannt werden, um sie in folgenden Bildern wiederzuerkennen. Die Idee hinter diesem Vorgehen ist eng mit menschlichem Verhalten in vergleichbaren Situationen verknüpft. Wenn ein Mensch sich in einer unbekannten Umgebung mithilfe visueller Reize orientieren soll, um beispielsweise aus einem Labyrinth zu entkommen, dann versucht er sich bestimmte Eigenheiten seiner Umwelt, z.B. Abzweigungen oder die Wegbreite, zu merken. Wenn er sich nun Stück für Stück durch das Labyrinth bewegt, so baut er in seinem Gehirn eine Karte von jenem auf und versucht seine Position darin zu bestimmen. In regelmäßigen Abständen blickt er sich erneut um, schätzt die zurückgelegte Strecke und ergänzt seine mentale Karte, je nachdem welche zuvor erkannten Merkmale er in der aktuellen Position wiedererkennt und welche zum ersten Mal. Dieses Vorgehen wird so lange wiederholt, bis ein Entkommen aus dem Labyrinth erfolgt.

An dieser Stelle muss noch einmal betont werden, dass mit „das zugrundeliegende Problem“ eigentlich zwei Probleme gemeint sind: Zum einen gilt es, die zu Beginn unbekannte Umgebung möglichst detailliert zu kartieren. Dies geschieht im Wesentlichen in Form von 3D-Raumpunkten, die zusammen ganze Oberflächen darstellen. Im Fall des Labyrinths würde dies die Position, Höhe und Dicke der Wände sowie der Verlauf und die Breite der Wege umfassen. Zum anderen muss die Position und Orientierung der Kamera, oder des Menschen, in dieser Umgebung geschätzt werden. Wäre zu Beginn die Umgebung genau bekannt, so ließe sich die Kamerapose relativ unkompliziert errechnen, da die im Bild erkannten Punkte im Regelfall leicht den Umgebungspunkten zugeordnet werden könnten. Eine reine Lokalisierung ließe sich also ohne allzu große Schwierigkeiten realisieren, genauso wie das Entkommen aus einem Labyrinth keine große Hürde darstellt, wenn ein Plan der verschiedenen Wege vorliegt. Stünde dagegen zu jedem Zeitpunkt eine verlässliche Information der Kamerabewegung und -ausrichtung zur Verfügung, so könnte diese genutzt werden, um die Kartierung der Umwelt wesentlich zu simplifizieren. Wenn man sich hier noch einmal das Labyrinth vorstellt, so liegt eine Schwierigkeit des „Herausfindens“ in der Ungenauigkeit, mit der der abgelaufene Pfad rekonstruiert wird. Bereits nach einigen Abzweigungen fällt das Wiederkennen von zuvor besuchten Abschnitten schwer, was zum einen der begrenzten Speicherfähigkeit des menschlichen Gehirns zugeschrieben werden kann, aber auch eine Folge von fehlerbehafteter Nachverfolgung der eigenen Bewegung ist.

Tatsächlich gilt es aber beide Aufgaben gleichzeitig zu lösen, wobei eine gegenseitige Beeinflussung offensichtlich ist. Wenn immer auf die geschätzte Karte vertraut wird, diese aber fehlerhaft ist, so wird auch das Tracking der Kamera ungenau werden und wenn immer die errechnete Kameraposition als richtig angenommen wird, dann wird sich eine Ungenauigkeit in dieser Berechnung, immer auch in der Karte niederschlagen. Anhand dieser Betrachtung lässt sich zudem leicht erkennen, dass nicht korrigierte Fehler sich selbst verstärken - denn die Ergebnisse zu einem Zeitpunkt sind die Grundlage für die Schätzung des nächsten Zeitschritts. Es muss also eine ständige Korrektur der Karte sowie der Pose auf Basis der jeweils anderen Information stattfinden, um zu einer zufriedenstellenden Lösung zu gelangen.

Bevor aber der Ablauf des Algorithmus noch genauer vorgestellt wird, muss ein anderes Problem untersucht werden: Da das Kamerabild die dreidimensionale Rauminformation in der zweidimensionalen Bildebene darstellt, ist offensichtlich, dass bei der dabei stattfindenden Projektion Tiefeninformation verloren geht. Folglich ist es nicht exakt möglich den Abstand von im Bild zu sehenden Objekten zur Kamera und ihre genaue Oberflächenstruktur aus nur einem Bild zu bestimmen. Deshalb muss zuerst geklärt werden, wie aus einer zweidimensionalen Bildinformation dreidimensionale Umgebungspunkte erzeugt werden.

2.2 Triangulation

Wie im vorherigen Abschnitt beschrieben, ist eine exakte Rekonstruktion der Umgebungspunkte vor allem aufgrund der fehlenden Tiefeninformation nicht mit einem einzelnen Bild möglich. Die Triangulation ist ein Verfahren, mit dem sich übereinstimmende Merkmale von Bildern aus verschiedenen Kameraperspektiven einem dreidimensionalem Raumpunkt zuordnen lassen. Diese Beschreibung enthält bereits zwei wichtige Punkte: Zum einen werden Merkmale betrachtet, die in mehreren Aufnahmen gefunden und erfolgreich in Übereinstimmung

gebracht werden konnten. Zum anderen müssen diese Aufnahmen aus verschiedenen Perspektiven sein, damit das Ergebnis genau ist. Warum diese Anforderungen wichtig sind und wie sie sich zur Berechnung des Umgebungspunktes nutzen lassen, soll anhand des grundsätzlichen Ablaufs erklärt werden.

2.2.1 Merkmalerkennung und -übereinstimmung

Zu Beginn wird ein einzelnes Bild mit dem Ziel betrachtet, Merkmale, auch Landmarker oder Features genannt, zu erkennen. In den klassischen Anwendungsfällen wie dem Autonomen Fahrzeug sind das zum Beispiel Häuserkanten, Straßenschilder oder Bordsteinkanten. Im Wesentlichen handelt es sich dabei also um Objekte, die sich von ihrer Umgebung durch Form, Farbe und Struktur abheben und im Bild entsprechend zu erkennen sind. Wie bereits in der Einleitung dieses Kapitels angemerkt, sind Lokalisierungsalgorithmen oder die Bildverarbeitung im Allgemeinen keine neuen Forschungsgebiete. Folglich gibt es eine Vielzahl von Verfahren, die eine robuste Merkmalerkennung ermöglichen. Da sich diese Arbeit im Wesentlichen mit den Problemen von verformbaren Umgebungen bei der Navigation beschäftigt, liegt das Hauptaugenmerk nicht auf der Merkmalerkennung und -beschreibung selbst. Der als Ausgangspunkt verwendete SLAM-Algorithmus benutzt für diese Zwecke die ORB(*Oriented FAST and Rotated BRIEF*)-Merkmalerkennnung, welche in [Rub+11] präsentiert wurde und sich auch in den getesteten Szenarien als zuverlässig erwiesen hat. ORB greift dabei auf das effiziente FAST-Verfahren [RD06] zur Erkennung und BRIEF-Deskriptoren [Cal+10] zur binären Beschreibung der Features zurück. Zudem wurden Effizienz-steigernde Maßnahmen sowie eine rotations-invariante Beschreibung ergänzt, was eine weite Verbreitung des Verfahrens zur Folge hat.

An dieser Stelle sei schonmal angemerkt, dass die Operationsumgebung andere Landmarker bereithält als die Ortschaft, durch die das Auto fährt. Hier fungieren beispielsweise Blutgefäße, die Gewebestruktur oder dessen potentielle Verfärbung als Orientierungshilfen. Sich hierdurch ergebende Probleme oder notwendige Anpassungen wurden für diese Arbeit nicht weiter betrachtet, da keine realen Endoskopie-Aufnahmen verwendet wurden. Klar ist, dass auch hier die Verlässlichkeit des verwendeten Algorithmus untersucht werden muss, da von ihr die Genauigkeit und Robustheit des gesamten Lokalisierungsverfahrens abhängt. Aufgrund des Fokus dieser Arbeit ist die klassische ORB-Detektion aber ausreichend und soll hier nicht weiter diskutiert werden.

Abbildung 2.1 zeigt eine Aufnahme des Instituts für Systemdynamik (ISYS) und die darin erkannten ORB-Punkte. Die hier abgebildeten Punkte sind nur 1000 der tatsächlich Gefundenen, gleichverteilt über das gesamte Bild. Dieser Prozess wird nun für ein weiteres Bild aus einer anderen Perspektive wiederholt. Das zwischen den beiden Aufnahmen eine ausreichende Kamerabewegung liegen muss, ist vor allem Präzisionsgründen geschuldet und wird im nächsten Abschnitt genauer erläutert. Nun werden Übereinstimmungen zwischen den gefundenen Features gesucht, also diejenigen, die sowohl in der ersten als auch in der zweiten Ansicht zu identifiziert wurden. Dafür werden die binären Beschreibungen von ORB genutzt, die im Wesentlichen die Umgebung des Bildpunktes, also z.B. des Hauskantenverlaufs, enthalten. Beispielhaft sind in Abbildung 2.2 ein weiteres Bild des ISYS und einige gefundene Übereinstimmungen dargestellt. Wichtig sind hier vor allem zwei Punkte: Die Zuordnung der Merkmalen in beiden Bildern ist nicht immer ganz akkurat. Es können Artefakte auftreten, so dass zwei Punkte verbunden werden, die nicht zum selben Raumpunkt gehören.

Abbildung 2.1: Aufnahme des ISYS und erkannte ORB-Merkmale.

Besonders anfällig hierfür sind sich wiederholenden Strukturen, wie die Fenster in der Abbildung. Gleichzeitig können tatsächliche Übereinstimmungen übersehen bzw. nicht als solche erkannt werden; diese werden folglich auch nicht dargestellt. Die Ursachen hierfür sind vielfältig, exemplarisch seien hier unterschiedliche Beleuchtung, Unschärfen im Bild oder durch die verschiedenen Perspektiven entstehende Unterschiede in der Erscheinung zu nennen.

2.2.2 Relativbewegung der Kamera und Berechnung des Raumpunkts

Bis hierhin wurden die beiden Bilder auf hervorstechende Kennzeichen untersucht und diese über die verschiedenen Perspektiven in Verbindung gebracht. Diese Information kann nun genutzt werden, um die relative Kamerabewegung zu approximieren. Ziel ist es also, die Translation und Rotation, die zwischen den beiden Posen liegen, zu bestimmen. Tatsächlich ist nur die Relativbewegung von Interesse, da der SLAM-Algorithmus in einem eigenen Koordinatensystem operiert. Die initiale Position kann also immer in den Ursprung gesetzt und die Achsen entsprechend der ursprünglichen Orientierung angepasst werden. Zwischen Umgebung und dem SLAM-Koordinatensystem kann durch die Verwendung homogener Koordinaten eine Transformation bestimmt werden, die auf einfache Art und Weise den Vergleich der Ergebnisse des Algorithmus mit Messdaten ermöglicht. Das hier beschriebene Vorgehen orientiert sich an den Methoden, die in [Zha98], [XZ13] und [Pol04] beschrieben werden.

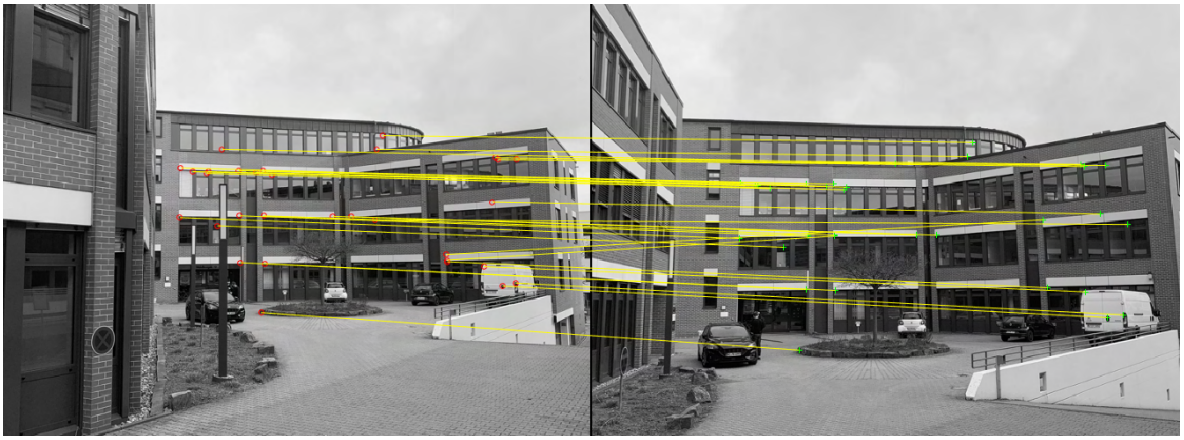

Abbildung 2.2: Gefundene Übereinstimmungen der ORB-Merkmale in den beiden Aufnahmen des ISYS.

Abbildung 2.3: Rekonstruktion des Raumpunkts mittels Triangulation.

Eine Kamerapose liegt in der Form

$$\mathbf{C}_i^{\mathrm{H}} = \begin{bmatrix} \mathbf{R}_{\mathrm{cam},i} & \overrightarrow{OP}_{\mathrm{cam},i} \\ 0^{1\times 3} & 1 \end{bmatrix}, \tag{2.1}$$

vor und enthält also sowohl die Position, in Form des Translations-Vektors $\overrightarrow{OP}_{\mathrm{cam},i} \in \mathbb{R}^{3\times 1}$, als auch Orientierung, gegeben durch die Rotationsmatrix $\mathbf{R}_{\mathrm{cam},i} \in \mathbb{R}^{3\times 3}$, der Kamera. Das hochgestellte H impliziert die Verwendung homogener Koordinaten. Die Matrix $\mathbf{C}_i^{\mathrm{H}} \in \mathbb{R}^{4\times 4}$ gibt also die Transformation vom Ursprung zur entsprechenden Pose an.

Wie zuvor angemerkt, kann die erste Pose

$$\mathbf{C}_1^{\mathrm{H}} = \begin{bmatrix} \mathbf{I}^{3\times 3} & 0^{3\times 1} \\ 0^{1\times 3} & 1 \end{bmatrix} \tag{2.2}$$

in den Ursprung des Weltkoordinatensystems gesetzt werden und enthält entsprechend die Einheitsmatrix $\mathbf{I}$ für die Rotation, Nullen für die Translation und eine Eins für die homogene Transformation. In der Verkettung berechnet sich die j-te Pose

$$\mathbf{C}_j^{\mathrm{H}} = \mathbf{T}_{ji}^{\mathrm{H}} \mathbf{C}_i^{\mathrm{H}} \tag{2.3}$$

immer über die Relativbewegung $\mathbf{T}_{ji}^{\mathrm{H}} \in \mathbb{R}^{4\times 4}$ gegenüber der i-ten Pose. Es soll nun untersucht werden, wie im Allgemeinen aus dem Übereinstimmungen der Merkmale zweier Aufnahmen bei gegebener Pose zum Zeitpunkt der ersten Aufnahme die zweite Kamerapose bestimmt werden kann.

Um die folgenden Schritte genauer zu verstehen, wird zuerst das verwendete Lochkameramodell, das als Basis der projektiven Geometrie dient, eingeführt. Dabei wird ein Raumpunkt $P = (X, Y, Z)$ dargestellt in Weltkoordinaten über den Brennpunkt in die Bildebene projiziert und entspricht dort dem Bildpunkt $p = (x, y)$ in Pixelkoordinaten. Die Projektion kann dabei als einfache Matrix-Vektor-Multiplikation dargestellt werden, aus der die zweidimensionale Pixelposition

$$\mu \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{KH} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

hervorgeht. Dabei ist μ ein Skalierungsfaktor, $\mathbf{K}$ enthält die intrinischen Kameraparameter, die die Ausrichtung der Bildebene bestimmen. Diese werden als bekannt vorausgesetzt und können im Voraus durch Kalibrierung der Kamera bestimmt werden. Hier sei erwähnt, dass die Genauigkeit der ermittelten Parameter großen Einfluss auf das Gesamtresultat hat und daher im Allgemeinen bei kamera-basierten Verfahren zu berücksichtigen ist. Eine kurze Erklärung der relevanten Parameter findet sich in Anhang A. Die Matrix $\mathbf{H}$ beinhaltet die extrinsischen Kameraparameter und gibt die Transformation von Welt- in Kamera-Koordinaten an. Dies ist notwendig, da $\mathbf{K}$ lediglich die Transformation von Kamera- zu Bildkoordinaten repäsentiert, der Umgebungspunkt aber in Weltkoordinaten vorliegt. Im betrachteten Fall folgt für Zeitschritt i

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{R}_{\mathrm{cam},i}^{-1} & -\mathbf{R}_{\mathrm{cam},i}^{-1} \overrightarrow{OP}_{\mathrm{cam},i} \end{bmatrix} \in \mathbb{R}^{3\times 4} \tag{2.5}$$

aus der Inversen der bekannten Kamerapose $\mathbf{C}_i^{\mathrm{H}}$. Die Matrix $\mathbf{H}_i$ stellt dabei allerdings keine homogene Transformation dar, sondern besteht lediglich aus der inversen Rotationsmatrix und

entsprechenden Rücktranslation der i-ten Kamerapose. Da $\mathbf{C}_i^{\mathrm{H}}$ genau die Transformation vom 3D-Kamerasystem der i-ten Aufnahme zu den 3D-Weltkoordinaten angibt, lässt sich dieser Zusammenhang leicht erkennen. Letztlich folgt aus dem Umgebungspunkt $P = (X, Y, Z)$ also der zweidimensionale Bildpunkt $p = (x, y)$ durch Multiplikation mit der Projektionsmatrix $\mathbf{M}$.

Wenn beispielhaft ein Merkmal, das in beiden Bildern gefunden wurde, betrachtet wird, so ist auch die jeweilige Pixelposition $(x^{\mathrm{a}}, y^{\mathrm{a}})$ in Aufnahme a bzw. $(x^{\mathrm{b}}, y^{\mathrm{b}})$ in Bild b bekannt. Dabei gilt

$$\begin{bmatrix} x^{\mathrm{b}} & y^{\mathrm{b}} & 1 \end{bmatrix} \mathbf{F} \begin{bmatrix} x^{\mathrm{a}} \\ y^{\mathrm{a}} \\ 1 \end{bmatrix} = 0 \tag{2.6}$$

mit der Fundamentalmatrix $\mathbf{F} \in \mathbb{R}^{3\times3}$ als Folge der Epipolargeometrie [Pol04]. Die Fundamentalmatrix

$$\mathbf{F} = \mathbf{K}^{-\top}\mathbf{E}\mathbf{K}^{-1} = \mathbf{K}^{-\top}\mathbf{R}[\mathbf{t}]_{\times}\mathbf{K}^{-1} \tag{2.7}$$

setzt sich aus den intrinsischen Kameraparametern und der Relativbewegung in Form der Essential-Matrix $\mathbf{E}$ zusammen. Diese beinhaltet die zwischen den beiden Posen liegende Rotation $\mathbf{R}$ und Translation $\vec{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^{\top}$, wobei

$$[\mathbf{t}]_{\times} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{2.8}$$

die zugehörige schief-symmetrische Matrix beschreibt, die das Kreuzprodukt durch eine Matrix-Vektor-Multiplikation ersetzt. Da $\mathbf{F}$ zwar neun Einträge hat, aber nur bis zu einem bestimmten Skalierungsfaktor bestimmt werden kann, besitzt sie acht Unbekannte, weswegen mindestens ebensoviele übereinstimmende Merkmale zur Berechnung mit einem linearen Verfahren benötigt werden. Durch Umsortierung kann Gleichung (2.6) in die Form

$$\begin{bmatrix} x^{\mathrm{a}}x^{\mathrm{b}} & y^{\mathrm{a}}x^{\mathrm{b}} & x^{\mathrm{b}} & x^{\mathrm{a}}y^{\mathrm{b}} & y^{\mathrm{a}}y^{\mathrm{b}} & y^{\mathrm{b}} & x^{\mathrm{a}} & y^{\mathrm{a}} & 1 \end{bmatrix} f = 0 \tag{2.9}$$

mit $f = \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{21} & F_{22} & F_{23} & F_{31} & F_{32} & F_{33} \end{bmatrix}$ gebracht werden. Für n Übereinstimmungen $(x_k^{\mathrm{a}}, y_k^{\mathrm{a}})$ und $(x_k^{\mathrm{b}}, y_k^{\mathrm{b}}), k \in [1, \mathrm{n}]$ können die Gleichungen zu

$$\mathbf{A}f = 0 \tag{2.10}$$

zusammengefasst werden - $\mathbf{A} \in \mathbb{R}^{\mathrm{n}\times9}$ enthält in jeder Reihe die Pixelinformation eines Merkmalpaars, wie der Reihenvektor in Gleichung (2.9). Das Gleichungssystem kann einfach mittels Singulärwertezerlegung gelöst und somit die Fundamentalmatrix aus f rekonstruiert werden [GVL13]. Anschließend kann die Essential-Matrix $\mathbf{E}$ über den Zusammenhang aus Gleichung (2.7) bestimmt werden. Um die Rotation und Translation zu berechnen, kann die Tatsache genutzt werden, dass für jeden Translationsvektor $\vec{t}$ eine orthonormale Matrix $\mathbf{V}$ existiert, so dass

$$[\mathbf{t}]_{\times} = \sigma'\mathbf{V}\mathbf{Z}\mathbf{V}^{\top} \tag{2.11}$$

mit $\sigma' = |\vec{t}|$ und einer schief-symmetrischen Matrix $\mathbf{Z} \in \mathbb{R}^{3\times3}$. Dieser Zusammenhang lässt sich zu $[\mathbf{t}]_{\times} = \sigma'\mathbf{V}\mathbf{W}\hat{\mathbf{D}}\mathbf{V}^{\top}$ mit einer Permutationsmatrix $\mathbf{W} \in \mathbb{R}^{3\times3}$ und der Diagonalmatrix

$$\hat{\mathbf{D}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

umformen. Durch Multiplikation mit $\mathbf{R}$ ergibt sich

$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times = \sigma' \mathbf{R}\mathbf{V}\mathbf{W}\hat{\mathbf{D}}\mathbf{V}^\top = \mathbf{U}\mathbf{D}\mathbf{V}^\top \tag{2.13}$$

mit der orthonormalen Matrix $\mathbf{U} = \mathbf{R}\mathbf{V}\mathbf{W}$ und der Diagonalmatrix $\mathbf{D} = \sigma'\hat{\mathbf{D}}$. Mittels Singulärwertezerlegung von $\mathbf{E}$ können die Matrizen $\mathbf{U}$, $\mathbf{D}$ und $\mathbf{V}$ und daraus mit Gleichung (2.11) der Translationsvektor $[\mathbf{t}]_\times$ bis auf einen Skalierungsfaktor σ berechnet werden. Die Rotationsmatrix ergibt sich zu

$$\mathbf{R} = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top. \tag{2.14}$$

Grundsätzlich ist die Wahl der schief-symmetrischen Matrix $\mathbf{Z}$ und der Permutationsmatrix $\mathbf{W}$ beliebig. Aufgrund der Tatsache, dass Rotation und Translation je drei Freiheitsgrade haben, die Skalierung aber nicht bestimmt werden kann, folgt, dass $\mathbf{E}$ fünf Freiheitsgrade hat. Daraus ergibt sich $det(\mathbf{E}) = 0$, also ist auch ein Singulärwert gleich null. Darüber hinaus folgt, dass die beiden verbleibenden Singulärwerte gleich sein müssen - sie können aufgrund der fehlenden Skala zu eins festgelegt werden, so dass $\hat{\mathbf{D}}$ die Form in Gleichung (2.12) annimmt. Es muss also lediglich sichergestellt werden, dass $\hat{\mathbf{D}} = \mathbf{W}^\top\mathbf{Z}$ erfüllt ist. In der Literatur wird dafür häufig

$$\mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

gewählt [WT00].

Mit der berechneten Rotation und Translation lässt sich die Relativbewegung

$$\mathbf{T}^{\mathrm{H}}_{ba} = \begin{bmatrix} \mathbf{R} & \sigma\vec{t} \\ 0^{1\times 3} & 1 \end{bmatrix} \tag{2.16}$$

aus Gleichung (2.3) bestimmen. Da die zur Aufnahme a gehörende Pose $\mathbf{C}^{\mathrm{H}}_{\mathrm{a}}$ als bekannt angenommen wird, lässt sich hiermit die zweite Pose $\mathbf{C}^{\mathrm{H}}_{\mathrm{a}}$ und mit (2.5) auch die zugehörige extrinsische Kameramatrix $\mathbf{H}_{\mathrm{b}}$ berechnen. Letztlich ergeben sich die Projektionsmatrizen $\mathbf{M}_{\mathrm{a}} = \mathbf{K}\mathbf{H}_{\mathrm{a}}$ und $\mathbf{M}_{\mathrm{b}} = \mathbf{K}\mathbf{H}_{\mathrm{b}}$.

Aus der perspektivischen Projektion sind zudem die unmittelbaren Zusammenhänge

$$x = \frac{X}{Z} \tag{2.17a}$$

und

$$y = \frac{Y}{Z} \tag{2.17b}$$

zwischen der Raum- und Bildinformation einen Punktes bekannt. Sie stellen eine Normalisierung über die Z-Richtung auf die Bildebene dar. Einsetzen von Gleichung (2.4) in (2.17) ergibt

$$x = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}, \tag{2.18a}$$

$$y = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}. \tag{2.18b}$$

Dabei bezeichnet m^{a}_{ij} das i,j-te Element der Projektionsmatrix $\mathbf{M}$ für die Kameraperspektive a gemäß Gleichung (2.4).

Verwendung der beiden bekannten Bildpunkte und Umformung in Matrix-Vektor-Form liefert

$$\underbrace{\begin{bmatrix} (x^{\mathrm{a}}m^{\mathrm{a}}_{31} - m^{\mathrm{a}}_{11}) & (x^{\mathrm{a}}m^{\mathrm{a}}_{32} - m^{\mathrm{a}}_{12}) & (x^{\mathrm{a}}m^{\mathrm{a}}_{33} - m^{\mathrm{a}}_{13}) & (x^{\mathrm{a}}m^{\mathrm{a}}_{34} - m^{\mathrm{a}}_{14}) \\ (y^{\mathrm{a}}m^{\mathrm{a}}_{31} - m^{\mathrm{a}}_{21}) & (y^{\mathrm{a}}m^{\mathrm{a}}_{32} - m^{\mathrm{a}}_{22}) & (y^{\mathrm{a}}m^{\mathrm{a}}_{33} - m^{\mathrm{a}}_{23}) & (y^{\mathrm{a}}m^{\mathrm{a}}_{34} - m^{\mathrm{a}}_{24}) \\ (x^{\mathrm{b}}m^{\mathrm{b}}_{31} - m^{\mathrm{b}}_{11}) & (x^{\mathrm{b}}m^{\mathrm{b}}_{32} - m^{\mathrm{b}}_{12}) & (x^{\mathrm{b}}m^{\mathrm{b}}_{33} - m^{\mathrm{b}}_{13}) & (x^{\mathrm{b}}m^{\mathrm{b}}_{34} - m^{\mathrm{b}}_{14}) \\ (y^{\mathrm{b}}m^{\mathrm{b}}_{31} - m^{\mathrm{b}}_{21}) & (y^{\mathrm{b}}m^{\mathrm{b}}_{32} - m^{\mathrm{b}}_{22}) & (y^{\mathrm{b}}m^{\mathrm{b}}_{33} - m^{\mathrm{b}}_{23}) & (y^{\mathrm{b}}m^{\mathrm{b}}_{34} - m^{\mathrm{b}}_{24}) \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0. \tag{2.19}$$

Um nun zum Umgebungspunkt zu gelangen wird der Eigenvektor $\vec{P}' = \begin{bmatrix} X' & Y' & Z' & W' \end{bmatrix}^\top$ zum kleinsten Eigenwert der quadratischen Matrix $\mathbf{Q}^\top\mathbf{Q}$ berechnet (*least square optimization*) [HS97]. Der Punkt berechnet sich dann zu $P = (X, Y, Z) = (\frac{X'}{W'}, \frac{Y'}{W'}, \frac{Z'}{W'})$. Wird dieses Vorgehen für alle gefundenen Übereinstimmungen wiederholt, ergibt sich die gewünschte Rekonstruktion der Szene [Pol04; XZ13]. In Abbildung 2.3 ist die Triangulation beispielhaft für einen Punkt dargestellt. Hier ergibt sich der Raumpunkt als Schnittpunkt der beiden Geraden durch die Kameraposition und das Merkmal in der Bildebene.

2.3 Ablauf des SLAM-Algorithmus

In Kapitel 2.2 wurde gezeigt, wie im Allgemeinen mittels Triangulation aus zwei Aufnahmen Umgebungspunkte rekonstruiert werden können. Der SLAM-Algorithmus verwendet dieses Verfahren wiederholt, um aus einer Sequenz von Bilddaten, eine umfangreichere Kartierung der Umwelt vorzunehmen. Dieser Prozess läuft, wie im Blockschaltbild in Abbildung 2.4 dargestellt, in mehreren Schritten ab.

Zunächst werden die ersten beiden Aufnahmen verwendet, um eine initiale Karte anzufertigen. Das Vorgehen entspricht hier genau dem der Triangulation aus Kapitel 2.3. Zusätzlich zur Karte liefert dieser Schritt die ersten beiden Kameraposen und stellt die Basis des gesamten Algorithmus dar. Ungenauigkeiten hier beeinträchtigen die Performance nachhaltig, da im Vergleich zu späteren Rechenschritten nur eine geringe Anzahl übereinstimmender Merkmale bekannt ist - ruckartige Bewegungen oder sich wiederholende Strukturen in den Bildern sorgen hier am ehesten für Schwierigkeiten.

Im Anschluss wird für jedes neue Bild zunächst das Tracking, also die Bestimmung der Kamerabewegung im Vergleich zum letzten Frame, vorgenommen. Dieser Schritt, die Lokalisierung, entspricht den ersten drei Schritten der Triangulation: Landmarker im neuen Bild werden identifiziert, bevor diese mit den Features der vorherigen Aufnahme verglichen werden. Mithilfe dieses Abgleichs wird nun die Relativbewegung berechnet.

Im letzten Schritt, dem „Local Mapping“, also der Kartierung der lokalen Umwelt, werden dann neue Umgebungspunkte über den Schnittpunkt der Geraden durch die Bildebene bestimmt und zur Menge der Kartenpunkte hinzugefügt. Hierfür werden wiederum die gefundenen Übereinstimmungen, die noch keinem Punkt zugeordnet sind, verwendet. Bei Matches, die bereits in einem früheren SLAM-Schritt gefunden wurden - also Merkmale, die bereits in mindestens zwei anderen Bildern sichtbar waren und in diesem Zeitschritt wieder erkannt wurden - wird die Position des Raumpunkts angepasst, falls sich hier eine zu große Abweichung ergibt. In weiteren Optimierungsschritten werden die Kameraposen sowie die Karte zusätzlich verfeinert, was hier jedoch nicht näher betrachtet wird.

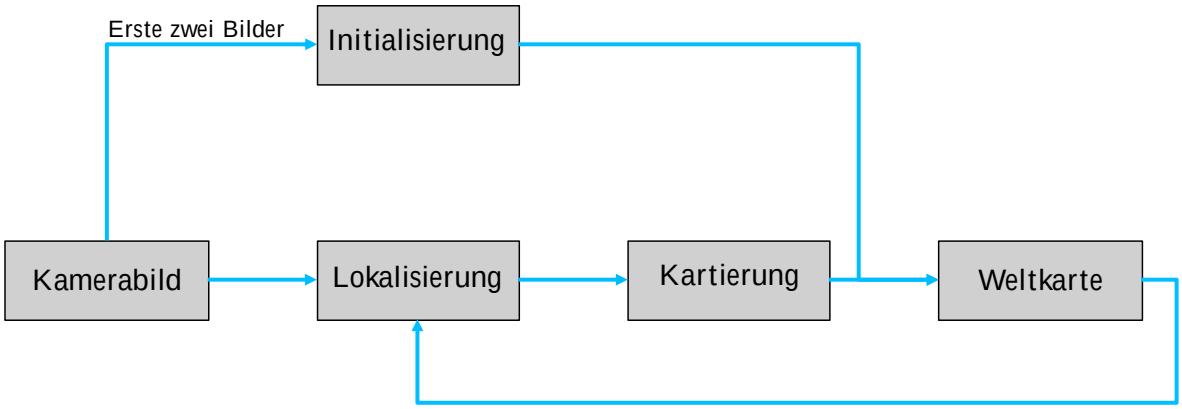

Abbildung 2.4: Ablauf des SLAM-Algorithmus: Nach der Initialisierung mit den ersten beiden Bildern, wird in jedem Schritt zunächst die Kamerapose errechnet und anschließend die Karte ergänzt. Dabei erfolgt die Lokalisierung auf Basis der aktuellen Weltkarte und den zuvor gefundenen Merkmalen.

Um die Rechenzeit zu verringern, wird der Ergänzung der Karte in manchen Fällen übersprungen und stattdessen direkt das nächste Bild verarbeitet. Als Entscheidungs-Variablen eignen sich hier beispielsweise der Betrag des Translationsvektors, der die Strecke zwischen den Aufnahmen beschreibt, oder die Anzahl an gefundenen Übereinstimmungen mit dem letzten Bild, auf dessen Basis die Karte aktualisiert wurde. Neben dem erwähnten Effizienzgewinn eignet sich diese Maßnahme zudem, um Fehler zu vermeiden, die durch unvorteilhafte Kamerabewegung entstehen. „Unvorteilhaft“ bezieht sich hier vor allem auf die zurückgelegte Strecke bzw. Rotation, die zwischen den beiden betrachteten Fotos liegt: Bei zu kurzen Distanzen wirken sich fehlerhafte Berechnungen der Kameraposen deutlich stärker auf die Genauigkeit der Kartenbildung aus. Da die Geraden, die für die Bestimmung der Umgebungsposition verwendet werden (siehe Abbildung 2.3) dann nahezu parallel verlaufen, ändert sich auch ihr Schnittpunkt maßgeblich. Allerdings muss sichergestellt werden, dass auch zu große Distanzen und damit einhergehend große Unterschiede in den Aufnahmen vermieden werden. Grund hierfür sind die verringerte Anzahl an Übereinstimmungen zwischen den Bildern und ihre richtige Zuordnung, die hauptverantwortlich für die Präzision des gesamten Verfahrens sind. Es gilt also in Abhängigkeit der Bildrate, Bewegungsgeschwindigkeit und Umgebung geeignete Parameter-Werte zu finden, so dass eine akkurate Lokalisierung und Kartierung ohne allzu lange Rechenzeiten möglich sind.

Als klassisches Verfahren wurde für diese Arbeit der ORB-SLAM, wie in [MMT15] vorgestellt, verwendet. In der zugehörigen Veröffentlichung ist der Ablauf des Algorithmus wesentlich detaillierter dargestellt, was aber für das Verständnis dieser Arbeit nicht notwendig ist. Stattdessen beschäftigt sich das nächste Kapitel mit notwendigen Erweiterungen dieses Ansatzes, um den Einsatz in verformbaren Umgebungen zu ermöglichen.

Kapitel 3

Modell-basierter SLAM-Algorithmus

Wie das vorherige Kapitel gezeigt hat, ermöglicht der SLAM-Algorithmus in starren Umgebungen eine zuverlässige Umgebungskartierung und Positionsbestimmung. Es stellt sich also die Frage, wieso dieser Ansatz ohne Anpassungen in der Operationsumgebung nicht angewandt werden kann und welche Erweiterungen von Nöten sind, um auch hier zu zufriedenstellenden Resultaten zu gelangen.

3.1 Grenzen des klassischen SLAM-Algorithmus in verformbaren Umgebungen

Um sich dieser Aufgabe zu nähern, soll hier nochmal an die Erklärungen aus Kapitel 1 erinnert werden, wo bereits erwähnt wurde, dass es während der Tumorentfernung zu gezielten, aber auch unbeabsichtigten Gewebedeformationen kommt. Wenn man sich den Ablauf des klassischen SLAM-Algorithmus vor Augen führt, so werden zu einem beliebigen Zeitschritt Merkmale in einem Bild gesucht und denen des vorherigen Bilds zugeordnet. Hierbei kommt es bereits zum ersten Problem: Durch die Verformung kann sich die Erscheinung des Merkmals ändern. Wenn dem so ist, hat das unter Umständen eine Nicht-Erkennung, mit großer Sicherheit aber vor allem eine fehlerhafte oder Nicht-Zuordnung zur Folge.

Darüber hinaus ergeben sich weitere Schwierigkeiten, die die Genauigkeit einschränken. Offensichtlich ändert sich durch die Deformation auch die Raumposition einiger bereits gefundener Merkmale. Das hat zwei wesentliche Konsequenzen: Erstens veraltet dadurch die bisherige Karte, da ein Teil der hier gespeicherten, geschätzten Umgebungspunkte eine neue Position eingenommen hat. Findet die Verformung nur in sehr geringem Ausmaß und über einen langen Zeithorizont statt, dann korrigiert der Algorithmus durch die implementierten Optimierungsschritte diese Abweichungen weitestgehend zuverlässig selbst, indem die 3D-Positionen der Weltpunkte laufend angepasst werden. Bei ausreichender Deformation reichen diese Modifizierungen allerdings nicht aus, so dass die Karte in der Folge nicht mehr mit der Umgebung übereinstimmt. Dieser Vorgang stellt bereits eine große Hürde dar, da somit eine der zwei wesentlichen Aufgaben nicht mehr zufriedenstellend erledigt werden kann. Wie in Kapitel 2.2 gezeigt, orientiert sich der Algorithmus aber nicht über die dreidimensionale Rauminformation eines Merkmals, sondern über die zugehörige zweidimensionale Position in der Bildebene. Dies geschieht durch den Vergleich des neuen mit dem zuletzt für die Kartierung verwendeten Bildes. Wenn nun nach der Verformung eine neue Aufnahme aus der Perspektive des Referenzbildes gemacht werden würde, so unterschiede sich diese vom ursprünglichen Bild. Das zweite wesentliche Problem ist also, dass sich neben der Erscheinung auch die Position

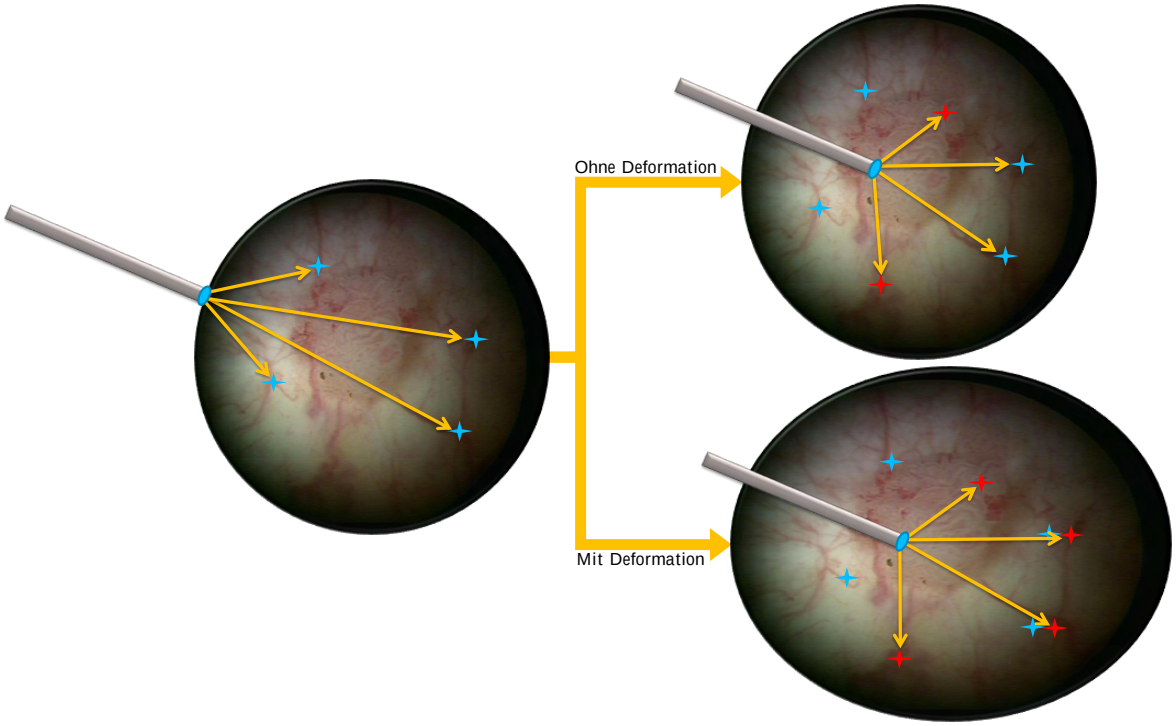

Abbildung 3.1: Schematische Darstellung der Probleme des klassischen SLAM-Algorithmus in verformbaren Umgebungen. Aus dem vorherigen Bild (links) sind Merkmale und die zugehörigen Umgebungspunkte bekannt. Wenn das System starr ist (rechts oben), werden die Punkte erfolgreich wiedererkannt und das Verfahren funktioniert wie gewünscht. Wenn es zu Verformung zwischen den Aufnahmen kommt (rechts unten), dann ist die gespeicherte Position der Merkmale und entsprechend auch die geschätzte Umgebungskarte veraltet. Zusätzlich werden eigentlich bekannte Merkmale nicht mehr als solche erkannt und die Lokalisierung wird ungenau.

von Merkmalen im Bild ändern kann. Infolgedessen ist die Zuverlässigkeit der Merkmalzuordnung eingeschränkt, was wiederum zu wesentlich ungenauerer Lokalisierung führt. Grund hierfür ist vor allem, dass sich die relativen Positionen der Merkmale im Bild ändern und folglich selbst bei richtiger Zuordnung keine einheitliche relative Kamerabewegung zwischen den Aufnahmen berechnet werden kann. Einheitlich bezieht sich hierbei auf ein alle Merkmalentsprechungen einschließendes Resultat. Um dennoch zu einem Ergebnis zu kommen, gibt es zwei Optionen. Naheliegenderweise kann auf eine Vielzahl von Übereinstimmungen vertraut werden, wobei jeweils eine Abweichung in Kauf genommen werden muss. Alternativ können einige der Übereinstimmungen stärker gewichtet werden, was wiederum die Fehleranfälligkeit erhöht, da diese Auswahl nicht eindeutig ist. Im ersten Fall wird der resultierende Fehler also eher gleichmäßig auf alle Merkmale aufgeteilt, während der zweite Fall nur konsistente Übereinstimmungen berücksichtigt und andere ausschließt. Unabhängig von diesen Überlegungen lässt sich die entstehende Abweichung nicht vollständig kaschieren und die Positionsbestimmung wird ungenau. Zur Verdeutlichung ist in Abbildung 3.1 schematisch der Vergleich zwischen starrer und verformbarer Umgebung dargestellt. Hierbei lässt sich die angeführte Problematik gut erkennen, da die Unterschiede in der Aufnahme deutlich zu sehen sind.

Die Anwendung des klassischen SLAM-Algorithmus zur Kartierung und Lokalisierung einer

verformbaren Umgebung resultiert in großen Ungenauigkeiten bei der Berechnung beider Aufgaben. Wie in Kapitel 2.1 angeführt, lassen sich diese beiden Effekte nicht trennen, wodurch Fehler sich mit fortschreitender Zeit selbst verstärken. Dieser Prozess wird durch weitere Verformungen beschleunigt, was große Abweichungen bei der Nachverfolgung der Kameratrajektorie und eine unzuverlässige Umgebungskarte nach sich zieht. Da diese Limitierungen seit geraumer Zeit bekannt sind, lassen sich in der Literatur viele verschiedene Ansätze finden. Als Basis dienen dabei oft die weit verbreiteten Systeme ORB-SLAM [MMT15] und Parallel Tracking and Mapping (PTAM) [KM07], aber auch [EKC17] befasst sich intensiv mit der Umgebungskartierung und Lokalisierung. Die Starrheits-Annahme, die diesen Ansätzen zugrunde liegt, wird dabei ausgenutzt, um akkurate und robuste Ergebnisse zu liefern. Entsprechend präsentieren [Lin+13] und [Wei+13] Erweiterungen der klassischen Verfahren, bei denen die gefundenen Merkmale in starre und nicht-starre getrennt werden, um so Verformungen entgegenzuwirken. Allerdings resultieren die genannten Techniken in spärlichen (*sparse*) Karten. Um eine dichtere Karte zu erlangen, werden in [QR18] künstliche Merkmale durch Laser Markierungen erzeugt, während in [Mah+18] mehrere Bilder zu Clustern angeordnet werden. Aufbauend auf ORB-SLAM befasst sich [Mah+16] mit dem Einsatz in der medizinischen Umgebung und liefert eine akkurate, semi-dichte Karte. Die Ergebnisse der Kameralokalisierung werden hier jedoch nicht validiert. Darüber hinaus sind die genannten Verfahren für kleine Verformungen ausgelegt und nur selten für große Deformationen getestet. Ein speziell für die minimal-invasive Chirurgie entwickelter Algorithmus ist MIS-SLAM [Son+18], dieser basiert aber auf dem Einsatz einer Stereo-Endoskopiekamera, was sich in der Praxis häufig als nicht gegeben erweist [Vel+16; Wag+12]. Darüber hinaus nutzt [Tur+17] eine RGB-D Kamera, so dass die fehlende Tiefeninformation des Bilds durch einen Tiefensensor ausgeglichen wird.

Um die Oberflächenstruktur aus den Aufnahmen einer einzelnen Kamera zu rekonstruieren, können Shape-from-Shading (SfS)-Ansätze genutzt werden, bei denen die Oberfläche mittels des sichtbaren Lichteinfalls erkannt wird [CB12; WNJ10; VSSY12]. Dafür wird neben einer externen Lichtquelle aber auch eine gleichbleibende, relative Pose zwischen Kamera und Lichtquelle benötigt, was sich im engen Umfeld minimal-invasiver Eingriffe als problematisch erweist. Weitere Methoden stellen Structure-from-Motion (SfM) und Shape-from-Template (SfT) dar. SfM berechnet die Deformation aus einer Reihe von Bildern, was zu guten Ergebnissen führt, aber aufgrund des hohen Rechenaufwands für Echtzeitanwendungen ungeeignet ist [Lin+16; Hao+15; MBC11]. SfT benötigt nur ein einzelnes Bild, dafür aber ein bekanntes 3D-Template in Form eines Modell mit Textur aus einer vorherigen Bildsequenz. Der in [Lam+21] präsentierte DefSLAM kombiniert das Sft-Verfahren aus [LM18] mit dem Non-rigid-Structure-from-Motion (NrSfM)-Verfahren aus [PPB17] und ist laut Autoren der erste echtzeitfähige, monokulare SLAM-Algorithmus für deformierbare Systeme.

Alle bestehenden Ansätze basieren auf der Rekonstruktion der Deformation aus dem Kamerabild, wenngleich hierfür verschiedene Techniken zum Einsatz kommen. Im Gegensatz dazu befasst sich diese Bachelorarbeit mit einem Modell-basierten Ansatz, welcher auf dem ORB-SLAM2 aufbaut [MAT17]. Die zentrale Idee des Modell-basierten Ansatzes ist es, die Verformung mittels Simulation vorherzusagen, die daraus gewonnenen Informationen zu nutzen und den klassischen Algorithmus darauf anzuwenden. Dieses Kapitel soll dazu dienen, dieses Konzept präziser zu formulieren und die Schritte, die zu seiner Realisierung notwendig sind, zu spezifizieren. Im Anschluss wird das hierzu implementierte Framework vorgestellt und die Funktionsfähigkeit des Schemas durch simulative Untersuchungen gezeigt.

3.2 Konzept eines Modell-basierten SLAM-Algorithmus

Um die Grundidee des Konzepts verständlich zu machen, sei hier an die Probleme erinnert, die sich bei Konfrontation des klassischen SLAM-Algorithmus mit Deformationen ergeben: Als Folge der veränderten Raumposition der Merkmale wird einerseits die Karte ungenau. Da sich andererseits auch die zugehörige Pixelposition in der Bildebene ändert, erfolgt die Merkmalzuordnung nicht mehr zuverlässig und auch die Lokalisierung der Kamera wird inakkurat. Der Modell-basierte Ansatz zielt darauf ab, diese beiden wesentlichen Komplikationen zu umgehen. Wie der Name impliziert, basiert das Konzept auf einem Umgebungsmodell, das simulativ verformt werden kann.

Das Blockschaltbild in Abbildung 3.2 zeigt den grundsätzlichen Ablauf und Informationsaustausch. Der blaue Block am linken Rand stellt die reale Umgebung dar, aus der zum einen das Kamerabild, zum anderen die auf die Umgebung wirkenden Kräfte gewonnen werden. Das heißt, dass Kenntnis der Kräfte eine Voraussetzung des vorgestellten Entwurfs darstellt. In der Praxis beweist sich diese Annahme teilweise als problematisch, da eine genaue Messung der Kräfte nicht möglich ist und daher andere Verfahren notwendig sind, um Aufschluss über ihre tatsächliche Wirkung zu erlangen. Im Fall der Blase ist es dafür aber möglich, das Volumen zu messen und darüber Rückschlüsse auf die Verformung zu ziehen. Diese Überlegungen werden im Rahmen der vorliegenden Arbeit nicht weiter diskutiert - stattdessen ist diese Thematik Inhalt zukünftiger Arbeiten. Setzt man Wissen über die Kräfte voraus, so hat der Modell-basierte SLAM-Algorithmus in Form des grau hinterlegten Blocks zwei Eingänge. In seinem Inneren befinden sich zwei Subsysteme, die sich gegenseitig beeinflussen und für die Ermittlung der beiden Ausgänge „Lokalisierung“ und „Kartierung“ zuständig sind. Die Ausgänge entsprechen also genau den Zielen, die der eigentlichen Aufgabenstellung zugrundeliegen. Ihre Darstellung mit gestrichelten Linien soll andeuten, dass sich die Zuordnung zu den beiden Blöcken innerhalb des grauen Blocks eher auf deren Aufgabenteilung als den tatsächlichen Informationsfluss bezieht. Wie ist das zu verstehen?

Der klassische SLAM-Algorithmus, der durch den grünen Block repräsentiert wird, erledigt für sich bereits beide Aufgaben. Wie Kapitel 2 gezeigt hat, ist das Verfahren zwar in der Lage, Kartierung und Lokalisierung gleichzeitig vorzunehmen, die Qualität der Ergebnisse in verformbarer Umgebung wird jedoch unzureichend sein. Zur Bewältigung dieser Aufgabe wird nun ein neuer Block eingeführt: Die rot hinterlegte Modell Prädiktion. Dieser Block erhält die auf das Objekt wirkenden Kräfte und bestimmt anhand dieser dessen Verformung. Auf Basis dieser Vorhersage wird nun die Umgebungskarte aktualisiert und dem SLAM-Algorithmus für die Schätzung der Position und Ergänzung der Karte mit dem nächsten Kamerabild zur Verfügung gestellt. Die Darstellung soll nun verdeutlichen, dass bei Kenntnis der Deformation der SLAM-Algorithmus vermehrt für die Lokalisierung zuständig ist, da die Karte mithilfe der Prädiktion aktuell gehalten wird. Klar ist jedoch, dass die Ergänzung der Karte nur durch den grünen Block geschehen kann, weswegen die abgebildete Arbeitsteilung nur Anschauungszwecken dient. Wie genau der Austausch zwischen den beiden Blöcken aussieht, werden die nächsten Kapitel zeigen.

In realen Systemen wird das Modell, welche Form es auch immer hat, nie genau der tatsächlichen Umgebung entsprechen. Das soll es auch gar nicht, schließlich beinhaltet die Bezeichnung per Definition bereits, dass es sich um eine vereinfachte Abbildung der Realität handelt. Um eine effiziente Simulation zu gewährleisten, müssen Beschränkungen in Hinblick auf die Komplexität des Modells akzeptiert werden. Als Resultat wird die errechnete Verformung nicht

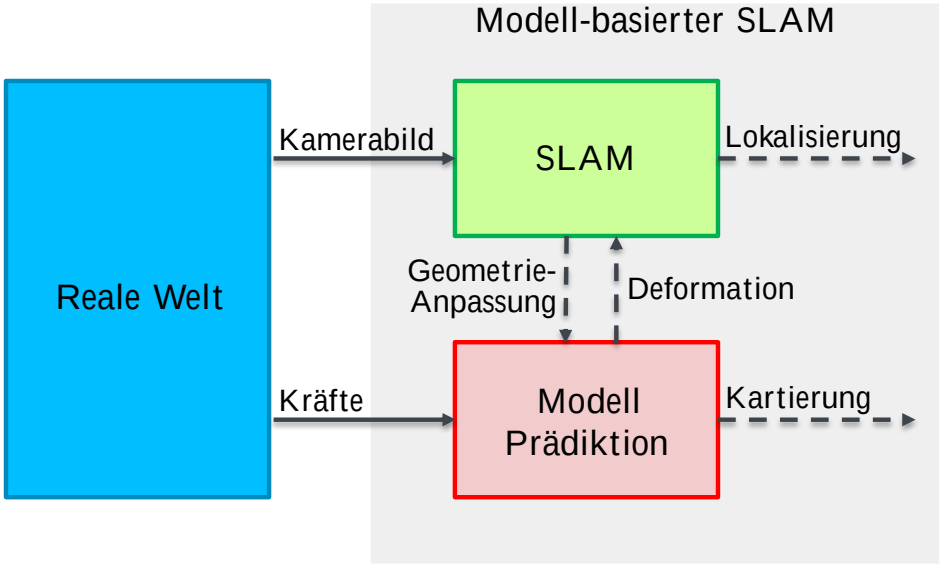

Abbildung 3.2: Blockschaltbild des Grundkonzepts des Modell-basierten SLAM-Algorithmus.

der tatsächlich beobachtbaren entsprechen und es gilt abzuwägen, ob auf die Vorhersage des Modells vertraut werden kann oder ob das Kamerabild und die mit dem klassischen Verfahren geschätzte Karte verlässlichere Informationen liefern. Dabei wird es zu zwei Unterscheidungen kommen: Im Regelfall wird das unveränderte Vorgehen des Algorithmus für betragsmäßig kleine Kräfte weiterhin recht gut funktionieren, da die Änderung von Bild zu Bild gering ist und die internen Optimierungsschritte die Verformung weitestgehend einbeziehen werden. Natürlich wird auch hier eine Verschlechterung im Sinne von größeren Abweichungen bei der Nachverfolgung der Kamerabewegung gegenüber dem starren Fall zu erkennen sein. Diese wird aber vergleichsweise gering ausfallen. Die hierbei durch das Modell hinzugewonnene Präzision wird sehr beschränkt sein, da die Ungenauigkeit des Modells stark ins Gewicht fällt. Es kann also sein, dass die durch den SLAM-Algorithmus erlangte Karte die Oberfläche genauer darstellt als das Modell es tut. Gegebenenfalls ist es dann sogar möglich, die Geometrie des simulierten Objekts entsprechend der Kartierung anzupassen und dadurch die Vorhersage zu verbessern. Wenn hingegen betragsmäßig große Kräfte wirken und es entsprechend zu einer deutlich sichtbaren Verformung kommt, wird der Modellfehler nur geringen Anteil haben und infolgedessen die mit dem Modell prädizierte Veränderung der Oberfläche zuverlässiger sein. Dann können die Vorhersagen genutzt werden, um die Performance des Algorithmus zu verbessern. Dieser Fall stellt also den in dieser Arbeit betrachteten dar - entsprechend wird im Folgenden vorausgesetzt, dass das verwendete Modell genau der Realität entspricht. Im Rahmen des Ausblicks in Kapitel 4 wird auf diese Thematik nochmal Bezug genommen.

Die vorherige Annahme - der genauen Kenntnis der wirkenden Kräfte - lässt sich mit der eben getroffenen - des realitätsnahen Modells - im Grunde zu einer einzigen vereinen. Dadurch kann die Hauptaufgabe, mit der sich dieses Kapitel beschäftigt, präziser formuliert werden. Vorausgesetzt die Verformung des betrachteten Objekts, also seine Oberflächenstruktur, sei genau bekannt, wie kann diese Information dann genutzt werden, um die Fehler des klassischen SLAM-Algorithmus zu korrigieren bzw. vorzubeugen? Um diese Frage weiter zu untersuchen, soll der nächste Abschnitt einen kurzen Einblick in die verwendete Simulationsumgebung liefern, die die Modell Prädiktion realisiert. Damit soll klar werden, in welcher Form die

Verformung überhaupt zur Verfügung steht, wie sie also genutzt werden kann. Gleichzeitig präsentiert dieses Unterkapitel in Kürze ein Framework, dass in realitätsnahen, medizinischen Simulationen Verwendung findet und somit auch in zukünftigen, auf dieser Arbeit aufbauenden Veröffentlichungen verwendet werden kann.

3.3 Modell Prädiktion und verwendetes Modell

Für die Modell Prädiktion wird die Simulation Open Framework Architecture (SOFA) verwendet. Diese Open Source Finite Elemente Methode (FEM)-Simulationsumgebung wurde 2004 von CIMIT und Inria entwickelt und wird vor allem für medizinische Simulationen und im Bereich Soft-Robotik eingesetzt. Ein wesentlicher Vorteil von SOFA ist die Verwendung von getrennten Modellen für die Simulation von Deformation, Kollision und Visualisierung, die in [Fau+12] ausführlich beschrieben ist. Dabei werden für jede der drei Betrachtungen separate Geometrien geladen, die sich in der Anzahl ihrer Knotenpunkte unterscheiden können. Anschließend werden diese drei Modell getrennt voneinander simuliert und durch Mappings miteinander verbunden. Folglich ist jedes der Modelle für eine eigene Aufgabe zuständig, ihre Konsistenz wird aber zu jedem Zeitpunkt gewahrt.

Das mechanische Modell fungiert dabei als Master und wird für die Berechnung der Verformung in Form der Positionen und Geschwindigkeiten der Knotenpunkte verwendet. Diese werden an das Kollisionsmodell übergeben, das nach jedem Zeitschritt bestimmt, ob das Objekt mit einem anderen zusammengestoßen ist und aus der Geschwindigkeit und Art des Aufpralls, die auf das Objekt wirkenden Kräfte bestimmt und wiederum an das mechanische Modell zur Berechnung der Deformation übergibt. Die Positionen der Knotenpunkte werden außerdem an das für die Visualisierung zuständige Modell übergeben, das vor allem die Oberflächengeometrie beinhaltet. Zudem ist es möglich, die Oberfläche um eine Textur zu erweitern, was der Simulation zum einen eine realistischere Optik verleiht, vor allem aber Rückschlüsse auf die sichtbare Verformung zulässt, die gerade für die in dieser Arbeit betrachteten Anwendungen hilfreich sein können.

Ein großer Vorteil von SOFA ist die durch die Verwendung der getrennten Modelle entstehende Möglichkeit die Genauigkeit der Simulation der verschiedenen „Bereiche“ (Mechanik, Kollision, Visualisierung) an die Anwendung anzupassen. Dadurch ermöglicht das Framework die effiziente Simulation von komplexen Szenen, wenn nur ein Aspekt dieser, beispielsweise die Kollision, wirklich von Interesse ist. SOFA ist echtzeitfähig - eine Tatsache, die gerade für das Operationsszenario durchaus relevant ist, wenn man bedenkt, dass der zeitliche Rahmen der Eingriffe eingeschränkt werden soll, wenn es um eine möglichst geringe Belastung des Patienten und im Allgemeinen um eine reibungslose Operation geht.

Konzeptionell soll mithilfe von präoperativen, bildgebenden Verfahren, wie z.B. MRT-Scans, ein initiales Modell des Organs erstellt werden, wobei das gesunde und befallene Gewebe unterschiedliche Materialeigenschaften zugewiesen bekommt. Die Modellierung des Gewebes und die Identifkation zugehöriger Parameter sind nicht Teil dieser Arbeit, sondern derzeitig Gegenstand weiterer Forschung. Abbildung 3.3 zeigt das im weiteren Verlauf verwendete Objekt, das zunächst beliebig gewählt wurde.

Die Oberfläche an sich besteht aus Knotenpunkten, die in Form von Dreiecken angeordnet sind. Jeder der Knotenpunkte kann sich frei im Raum bewegen, aus der Simulation kann

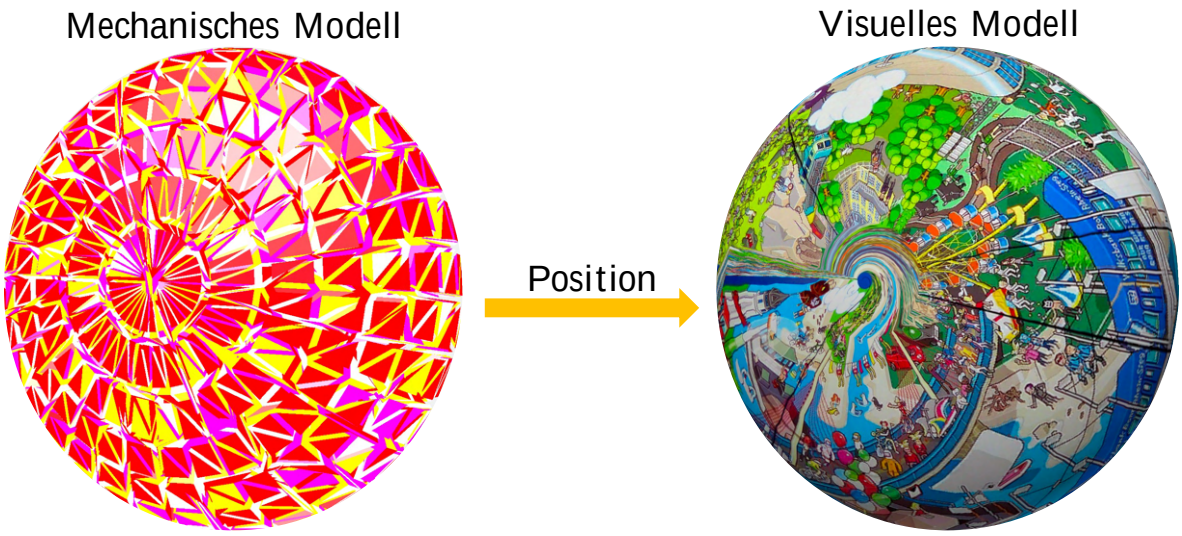

Abbildung 3.3: Darstellung des in der Simulation verwendeten Objekts. In SOFA lassen sich getrennte Modelle für die Mechanik (links) und Visualisierung (rechts) verwenden. Dabei fungiert das mechanische Modell als Master und berechnet die Verformung in Form der Position der Knotenpunkte des Meshs. Das visuelle Modell kann eine Textur in Form von Verfärbung oder, wie hier dargestellt, eines Bilds beinhalten. Grundsätzlich ist es möglich verschiedene Auflösungen für die Meshs der unterschiedlichen Modelle zu verwenden, um die Recheneffizienz zu erhöhen.

letztlich genau diese Position bestimmt werden. Zudem sind die Dreiecke in Form einer Liste, bei der jeder Eintrag aus den drei Indizes der Eckpunkte eines Dreiecks besteht, bekannt. So kann aus der Position der Knotenpunkte die genaue Oberflächenstruktur wiedergewonnen werden.

3.4 Berücksichtigung der Verformung

Nachdem das vorherige Unterkapitel sich vor allem mit der Simulation an sich beschäftigt hat und zeigen sollte, wie die Informationen über die Verformung gewonnen werden kann und in welcher Form sie letztlich zur Verfügung steht, soll nun das Einbeziehen dieser Kenntnisse zur Verbesserung des SLAM-Algorithmus betrachtet werden.

Wenn jeweils ein Schritt des SLAM-Algorithmus und der SOFA-Simulation ausgeführt wird, stehen im Anschluss folgende Informationen zur Verfügung:

- Die Menge der geschätzten Raumpunkte, die der Umgebungskarte entspricht
- Die Menge der geschätzten Kameraposen
- Eine Liste, die beinhaltet, welche Punkte aus welcher Kameraperspektive gesehen wurden
- Die zu jedem Raumpunkt gehörigen zweidimensionalen Pixelpositionen für jedes Bild, in dem der Punkt erkannt wurde

- Die prädizierte Oberfläche des Objekts als Menge von Knotenpunkten und zugehörigen Dreiecks-Verbindungen

Im Folgenden gilt es zwei Aufgaben zu erledigen:

- Sicherstellen, dass die Raumpunkte auf der Oberfläche des Objekts liegen und diese im weiteren Verlauf nicht verlassen. Dies entspricht einer ständigen Aktualisierung der Karte.
- Eine zuverlässige Merkmalzuordnung zu gewährleisten, so dass die Lokalisierung weiterhin präzise funktioniert.

Die nächsten beiden Unterkapitel widmen sich der Bewältigung der ersten Aufgabenstellung, bevor das dritte Unterkapitel einen Lösungsansatz für das zweite Problem präsentiert.

3.4.1 Projektion auf die Oberfläche

Zunächst soll also sichergestellt werden, dass die Menge der geschätzten Umgebungspunkte nicht beliebig im Raum, sondern auf der Oberfläche des Objekts liegen. Da die Oberfläche aus einer Vielzahl von Dreiecken besteht, gilt es, jeden einzelnen Punkt in die durch eines der Dreiecke aufgespannte Ebene zu projizieren.

Zuvor muss also geklärt werden, wie aus der Menge an Dreiecken dasjenige ausgewählt wird, das sich am besten für die Projektion eignet. Der projizierte Punkte muss letztlich innerhalb des entsprechenden Dreiecks liegen. Betrachtet man einen einzelnen Punkt P, so wird zunächst die Distanz zu jedem Knotenpunkt des Objekts in Form des Betrags des zugehörigen Verbindungsvektors betrachtet und der nächstgelegene Knotenpunkt ausgewählt. Im Anschluss wird die Menge an Dreiecken $\mathcal{D}$ bestimmt, die diesen Knotenpunkt beinhaltet. Jedes Element dieser Menge besteht dabei seinerseits aus den drei Eckpunkten eines Dreiecks, wobei es sich bei einem immer um den zuvor bestimmten, nähesten Knotenpunkt handelt.

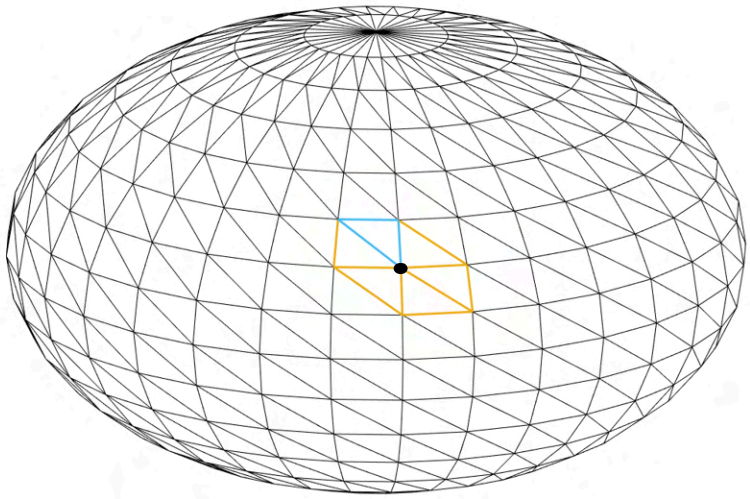

Abbildung 3.4: Aus der Menge an Dreiecken $\mathcal{D}$, die den Knotenpunkt beinhalten, der die kürzeste Distanz zum untersuchten Punkt aufweist, werden nacheinander die Dreiecke für die Projektion ausgewählt.

Die Idee hierbei ist, dass im Voraus nicht genau bestimmt werden kann, welches Dreieck tatsächlich geeignet ist, da die Position des projizierten Punktes wesentlich von der Kameraposition, aus der er erkannt wurde, abhängt. Betrachtet man die eben eingeführte Menge an Dreiecken $\mathcal{D}$, so schließen diese die gesamte umliegende Fläche ein. Wenn man nun davon ausgeht, dass der SLAM-Algorithmus grundsätzlich gut funktioniert, dann sind auch die geschätzten Punkte in den meisten Fällen nah an ihrer tatsächlichen Position. Da es sich zunächst nicht um ein verformtes System handelt, ist durchaus davon auszugehen, dass diese Annahme als gegeben angesehen werden kann. Schließlich ist die Funktionsweise und Zuverlässigkeit in starren Umwelten im Regelfall sichergestellt.

Abbildung 3.5: Schematische Darstellung der Projektion des geschätzten Punktes P auf die durch die Punkte A, B und C aufgespannte Ebene. Der projizierte Punkt P_{proj} entspricht dem Schnittpunkt der Ebene mit der Geraden, die durch die Kameraposition P_{cam}, aus der der Punkt erkannt wurde, und den geschätzten Punkt selbst verläuft. Da jeder Punkt aus mehreren Perspektiven erkannt wird, ergeben sich aus den verschiedenen, zugehörigen Kamerapositionen auch unterschiedliche Projektionen. Als Entscheidungskriterium dient hier der Schnittwinkel, zwischen Ebene und Gerade, der möglichst orthogonal sein soll.

Nachdem die Menge $\mathcal{D}$ bestimmt ist, wird nun nacheinander jedes der Dreiecke ausgewählt, bis eine passende Projektion gefunden wurde. Abbildung 3.4 zeigt für einen Knotenpunkt die Menge $\mathcal{D}$, aus der ein Dreieck herausgenommen wird. Zuerst wird der geschätzte Punkt P unter Verwendung der Kameraposition P_{cam} auf die durch die Eckpunkte A, B und C des Dreiecks aufgespannte Ebene E projiziert. Diese Projektion wird dabei einfach durch den

Schnittpunkt der Gerade

$$g : \vec{x} = \vec{p} + \lambda\vec{u}, \tag{3.1a}$$

$$\lambda \in \mathbb{R}, \quad \vec{p} = \overrightarrow{OP_{\text{cam}}}, \quad \vec{u} = \overrightarrow{P_{\text{cam}}P} \tag{3.1b}$$

durch die Kameraposition und den geschätzten Punkt mit der Ebene

$$E : \langle(\vec{x} - \vec{a})\vec{n}\rangle = 0, \tag{3.2a}$$

$$\vec{a} = \overrightarrow{OA}, \quad \vec{v_0} = \overrightarrow{AB}, \quad \vec{v_1} = \overrightarrow{AC}, \quad \vec{n} = \vec{v_0} \times \vec{v_1}. \tag{3.2b}$$

realisiert.

Der Schnittpunkt ist gegeben durch

$$\overrightarrow{OP_{\text{proj}}} = \vec{p} + \lambda_{\text{proj}}\vec{u} \tag{3.3a}$$

mit

$$\lambda_{\text{proj}} = \frac{(\langle\vec{a}, \vec{n}\rangle - \langle\vec{p}, \vec{n}\rangle)}{\langle\vec{u}, \vec{n}\rangle}, \tag{3.3b}$$

was sich durch einfaches Einsetzen und Umformen ergibt.

Nun gilt es zu klären, ob der resultierende, projizierte Punkt innerhalb des Dreiecks liegt, da er nur dann als valide angesehen werden kann. Andernfalls ist er zwar Teil der Ebene, aber nicht des Objekts - da aber letzteres von Interesse ist, ist die relative Lage des Punkte zum Dreieck entscheidend. Bisher wurde der Raumpunkt lediglich über seine Koordinaten im dreidimensionalen Raum beschrieben. Tatsächlich lässt diese Darstellung aber keine Aussage zur Relativposition zu, weshalb eine alternative Repräsentation des Punktes benötigt wird.

Hier bietet sich die Verwendung baryzentrischer Koordinaten an. Dabei wird der Punkt durch drei Parameter α, β und γ festgelegt, die als Gewichte in den Eckpunkten des Dreiecks interpretiert werden können und so gewählt sind, dass P_{proj} sich als Schwerpunkt des Dreiecks ergibt. In der Folge ergibt sich die alternative Darstellung

$$P_{\text{proj}} = \alpha A + \beta B + \gamma C, \tag{3.4}$$

wobei

$$v_2 = \overrightarrow{AP_{\text{proj}}}, \tag{3.5a}$$

$$\beta = \frac{\langle v_1, v_1\rangle\langle v_2, v_0\rangle - \langle v_0, v_1\rangle\langle v_2, v_1\rangle}{\langle v_0, v_0\rangle\langle v_1, v_1\rangle - \langle v_0, v_1\rangle\langle v_0, v_1\rangle}, \tag{3.5b}$$

$$\gamma = \frac{\langle v_0, v_0\rangle\langle v_2, v_1\rangle - \langle v_0, v_1\rangle\langle v_2, v_0\rangle}{\langle v_0, v_0\rangle\langle v_1, v_1\rangle - \langle v_0, v_1\rangle\langle v_0, v_1\rangle}, \tag{3.5c}$$

$$\alpha = 1 - \beta - \gamma. \tag{3.5d}$$

gilt. Der Punkt liegt innerhalb des Dreiecks, genau dann wenn

$$\alpha, \beta, \gamma \in [0, 1] \tag{3.6}$$

gilt. Die Berechnung mittels der Skalarprodukte und die einfache Überprüfung der relativen Lage erlauben dabei eine effiziente Implementierung, was die Verwendung baryzentrischer Koordinaten unterstützt [Eri05].

Abbildung 3.5 zeigt den schematischen Ablauf der Projektion und verdeutlicht eine weitere Schwierigkeit, die berücksichtigt werden muss. Aufgrund der Tatsache, dass jeder vom SLAM-Algorithmus geschätzte Punkt aus mindestens zwei verschiedenen Kameraperspektiven erkannt wird, folgt, dass zu jedem Punkt P mehrere Punkte auf der Oberfläche gefunden werden. Dabei kann sich die Position innerhalb eines Dreiecks stark unterscheiden und sogar ein anderes Dreieck für die Projektion als geeigneter erweisen. Die Entscheidung, welcher projizierte Punkt tatsächlich akzeptiert wird, kann auf vielfältige Art und Weise getroffen werden. Für diese Arbeit wurde der Schnittwinkel zwischen der Gerade und der Ebene als Kriterium ausgewählt, wobei möglichst orthogonale Schnitte bevorzugt werden. Dem zugrunde liegt die Idee, dass die laterale Position aus dem Kamerabild insgesamt leichter bestimmt werden kann, hier also nur geringe Abweichungen zu erwarten sind. Im Gegensatz dazu stellt die Rekonstruktion der Tiefe, die bei der Aufnahme verloren gegangen ist, eine größere Herausforderung dar. Zur Berechnung wird der Schnittwinkel

$$\phi_i = \arccos\left(\frac{\langle \vec{u_i}, \vec{n} \rangle}{\|\vec{u_i}\|\|\vec{n}\|}\right) \tag{3.7}$$

zwischen dem Richtungsvektor der Geraden der i-ten Kameraposition und dem Normalenvektor der Ebene untersucht. Da die beiden Vektoren in entgegengesetzte Richtungen zeigen, erfolgt die diskrete Wahl des projizierten Punktes

$$P_{\text{proj}} = P_{\text{proj},i^*} \tag{3.8}$$

über

$$i^* = \arg\min_i |\phi_i - \pi|. \tag{3.9}$$

Zusammenfassend lässt sich also sagen, dass durch einen einfachen Schnitt von Gerade und Ebene, alle vom SLAM-Algorithmus geschätzten Punkte, auf die durch die Simulation prädizierte Oberfläche projiziert werden. Dafür wird jeweils eine Menge von in der Nähe befindlichen Dreiecken ausgewählt und diese nacheinander überprüft. Sobald eine valide Projektion gefunden wird, wird dieser Prozess für die nächste Kameraposition wiederholt. Ergibt sich für diese ein Schnittwinkel, der näher an 90 Grade liegt als bei der vorherigen Projektion, so wird dieses Ergebnis als neuer Maßstab verwendet. Dabei werden jeweils die dreidimensionale Position und die baryzentrischen Koordinaten eines Punktes zusammen gespeichert.

3.4.2 Vorwärts-Prädiktion

Das Resultat des eben beschriebenen Vorgehens ist also die Menge an projizierten Punkten, die auf der Oberfläche des simulierten Objekts liegen. Wird das System nun weiter verformt, kommt es zu einer weiteren Änderung der Position eines Merkmals bzw. Oberflächenpunktes. Es gilt also über die Projektion hinaus sicherzustellen, dass die Karte auch bei zusätzlicher Deformation an die Simulation angepasst und somit aktuell gehalten wird. Auch aus diesem Grund wurden baryzentrische Koordinaten zur Beschreibung der projizierten Punkte gewählt. Wie im vorherigen Unterkapitel angeführt, beschreiben diese die relative Lage des Punktes innerhalb des Dreiecks, in das er projiziert wurde. Weil sich durch Verformung nicht die Zusammensetzung der Dreiecke, sondern lediglich die Positionen der Eckpunkte ändert, erweisen sich die baryzentrischen Koordinaten als unveränderlich gegenüber Deformationen.

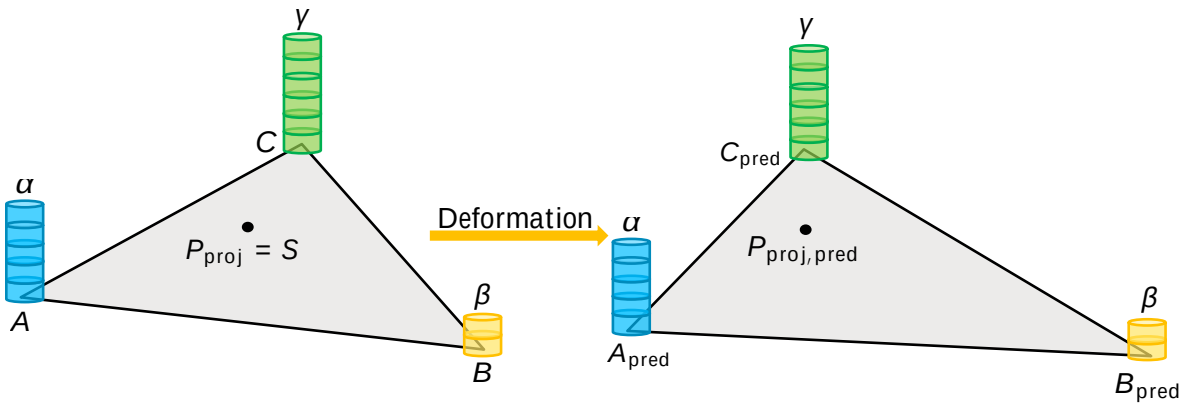

Abbildung 3.6: Schematische Darstellung der Vorwärts-Prädiktion. Links ist das Dreieck mitsamt dem bereits in die Ebene projizierten Punkt zu sehen. Zudem sind die baryzentrischen Koordinaten als Gewichte in den Knotenpunkten des Dreiecks dargestellt - diese sind so gewählt, dass P_{proj} gleichzeitig der Schwerpunkt ist. Nach der Simulation der Verformung stehen die neuen Positionen der Eckpunkte zur Verfügung. Diese können zusammen mit den baryzentrischen Koordinaten genutzt werden, um die neue Position $P_{\text{proj,pred}}$ des Punktes zu bestimmen.

Sind nach einem Zeitschritt, während dem es zu Verformung kommt, durch die gleichzeitige Simulation die prädizierten Dreieckspunkte A_{pred}, B_{pred} und C_{pred} bekannt, kann die neue Position des zuvor projizierten Punktes durch

$$P_{\text{proj,pred}} = \alpha A_{\text{pred}} + \beta B_{\text{pred}} + \gamma C_{\text{pred}} \tag{3.10}$$

bestimmt werden.

Nach einmaliger Projektion auf die Oberfläche, kann die Position im weiteren Verlauf des Algorithmus also einfach über die simulierte Verformung vorhergesagt werden, während die baryzentrischen Koordinaten unverändert bleiben. Es werden also lediglich neu erkannte Punkte projiziert, während die Position bestehender, bekannter Punkte bei weiterer Deformation aktualisiert wird, so dass zu jedem Zeitpunkt alle Kartenpunkte auf der Oberfläche des Objekts, genauer gesagt innerhalb eines der Dreiecke, liegen. Eine beispielhafte Darstellung dieser Vorwärts-Prädiktion für einen einzelnen Punkt ist in Abbildung 3.6 zu sehen. Hier lässt sich erkennen, dass die Position der Dreieckspunkte verändert wird und dadurch auch das Dreieck selbst eine neue Form erhält. Die Position des Punktes ändert sich folglich auch, kann aber auf einfache Weise mit Gleichung (3.10) berechnet werden.

3.4.3 Rückprojektion auf die Bildebene

Mit den Überlegungen der vorherigen beiden Abschnitte lässt sich das erste Problem, das sich in verformbaren Umgebungen ergibt, durch die Verwendung der simulierten Deformation, lösen: Die Karte als Menge von erkannten, dreidimensionalen Umgebungspunkten wird aktuell gehalten und entspricht durch die Projektion sogar eher der tatsächlichen Oberfläche des betrachteten Körpers, als dies nur durch den Einsatz des klassischen SLAM-Algorithmus möglich wäre. Wie in Kapitel 2 erläutert, nutzt letzterer aber nicht die Umgebungskarte, sondern die Merkmale in der Bildebene zur Orientierung. Bei ausreichender Verformung ist

die Funktionsweise der Lokalisierung also weiterhin eingeschränkt. Diese Problematik soll nun umgangen werden.

Abbildung 3.7 zeigt die zuvor verwendete Aufnahme des ISYS zunächst vor der Deformation. Wenn der Algorithmus bereits initialisiert wurde, steht aus den vorherigen Berechnungschritten die bisherige Karte und die zugehörige Kameratrajektorie zur Verfügung. Innerhalb des ersten Bildes kann der klassische SLAM-Algorithmus dann Merkmale erkennen, sie mit der vorherigen Aufnahme in Verbindung bringen und damit sowohl die Karte als auch die Kamerabewegung ergänzen. Wenn es nun während des darauffolgenden Zeitschritts zur Verformung kommt, dann wird die Aufnahme aus der nächsten Kameraposition auch diese verformte Umgebung zeigen. Um sich hieraus ergebende Fehler bei der Merkmalzuordnung zu vermeiden, wird versucht, die Positionen der Merkmale, die bei erneuter Aufnahme aus der vorherigen Kamerapose entstehen würden, zu rekonstruieren. Dieses Vorgehen entspricht im Optimalfall einer zweiten Aufnahme der nun verformten Umgebung aus der ersten Kameraperspektive, wie in der Abbildung dargestellt.

Für die Berechnung wird wiederum die aus den vorherigen Schritten bekannte 3D-Position des Punktes $P_{\text{proj,pred}}$ verwendet. Dieser wurde zunächst mittels Triangulation vom SLAM-Algorithmus geschätzt, dann auf die Oberfläche des simulierten Objekts projiziert und nach zusätzlicher Verformung vorwärts prädiziert. Er enthält daher bereits Information über die Deformation. Nun erfolgt eine Rückprojektion auf die Bildebene, deren Position, Ausrichtung und zugehörige Kamerapose in Form der Projektionsmatrix M bekannt ist. Die neue, prädizierte Position p_{pred} des Merkmals in 2D-Pixelkoordinaten kann nach Gleichung (2.18) durch

$$x_{\text{pred}} = \frac{m_{11}X_{\text{proj,pred}} + m_{12}Y_{\text{proj,pred}} + m_{13}Z_{\text{proj,pred}} + m_{14}}{m_{31}X_{\text{proj,pred}} + m_{32}Y_{\text{proj,pred}} + m_{33}Z_{\text{proj,pred}} + m_{34}}, \tag{3.11a}$$

$$y_{\text{pred}} = \frac{m_{21}X_{\text{proj,pred}} + m_{22}Y_{\text{proj,pred}} + m_{23}Z_{\text{proj,pred}} + m_{24}}{m_{31}X_{\text{proj,pred}} + m_{32}Y_{\text{proj,pred}} + m_{33}Z_{\text{proj,pred}} + m_{34}}. \tag{3.11b}$$

bestimmt werden. Wird dieses Vorgehen für alle im vorherigen Bild erkannten Punkte wiederholt, ergibt sich eine Rekonstruktion der verformten Aufnahme, die für das Finden der Merkmalübereinstimmungen besser geeignet ist. Wichtig ist zu betonen, dass die ORB-Merkmale nicht nur durch ihre Position in der Bildeben festgelegt sind, sondern durch Umgebungsdeskriptoren beschrieben werden, die für die Zuordnung essentiell sind. Hier wird angenommen, dass die Beschreibung der Merkmale unverändert bleibt, im Gegensatz zur Position sich ihre Erscheinung also nicht ändert. Die Rückprojektion auf die Bildebene stellt also eine intuitive Maßnahme dar, die Merkmalzuordnung und als Konsequenz auch die Lokalisierung der Kamera trotz Deformation zu ermöglichen.

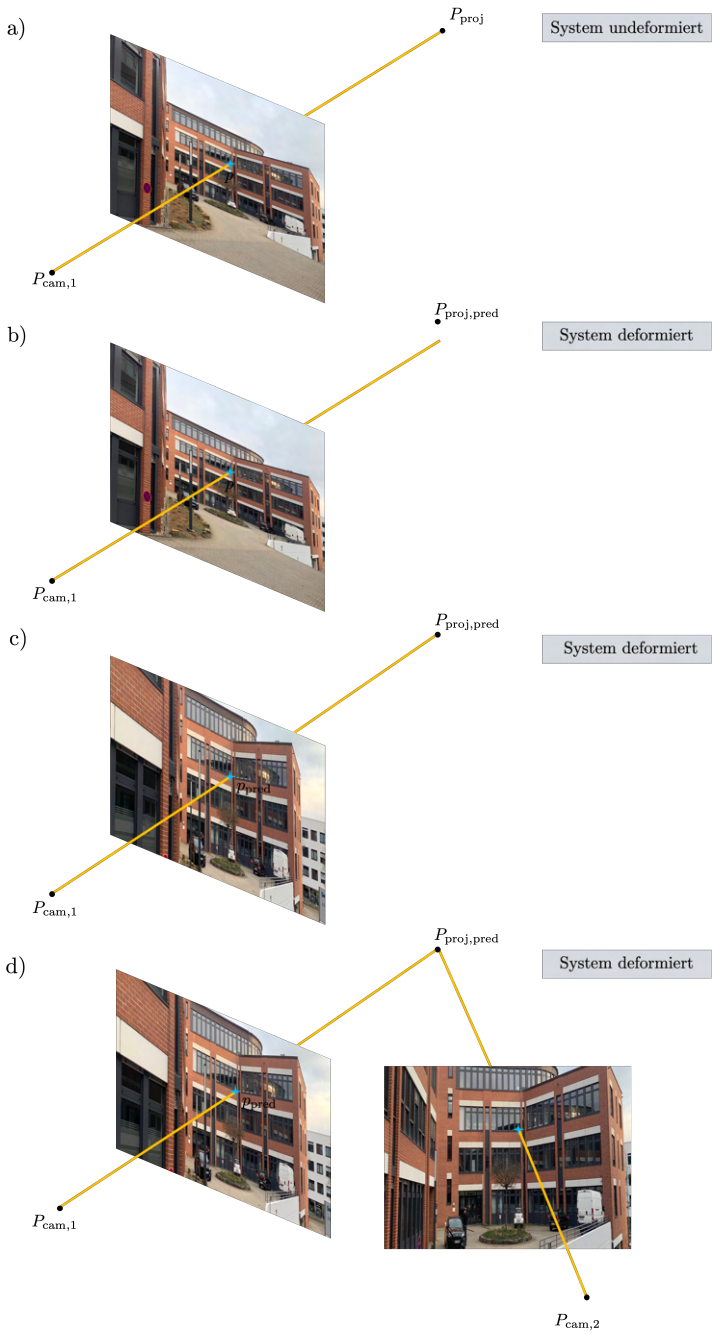

Abbildung 3.7: Idee der Rückprojektion in die Bildebene: a) Im unverformten System wird einem Merkmal im Bild ein Umgebungspunkt zugeordnet. b) Wenn sich infolge der Verformung die Raumposition des Punktes ändert, stimmt auch die zugehörige Pixelposition nicht mehr. c) Es erfolgt eine Rückprojektion des prädizierten Umgebungspunktes in die Bildebene. Wird dieses Vorgehen für eine Vielzahl von Punkten wiederholt, ergibt sich idealerweise eine Art Rekonstruktion einer Aufnahme der verformten Umgebung aus der gleichen Perspektive. Dieses Bild steht so nicht zur Verfügung, sondern ist hier nur zur Anschauung hinzugefügt. d) Wenn die Vorhersage genau ist, dann erfolgt die Merkmalzuordnung mit dem nächsten Bild wie in starren Umgebungen.

3.5 Framework und Implementierung

Kapitel 3.4 hat einen Ansatz präsentiert, mit dem sich die beiden wesentlichen Problemstellungen, mit denen sich diese Arbeit befasst, lösen lassen. Der Ablauf dieses Vorgehens wird durch Abbildung 3.8 nochmal verdeutlicht. In Kapitel 3.2 wurde anhand des übergeordneten Ablaufsplans bereits die Grundidee des Modell-basierten Konzepts vorgestellt. Die Methoden zur Projektion, Vorwärts-Prädiktion und Rückprojektion ergänzen das in Abbildung 3.2 eingeführte Blockschaltbild um einen weiteren Block, der die Informationen des klassischen SLAM-Algorithmus mit der Modell Prädiktion zusammenführt. Abbildung 3.9 zeigt das erweiterte Blockschaltbild. Die verbesserte Karte und die Anpassungen der Merkmalpositionen als Ausgang des gelben Blocks werden wieder an den SLAM-Algorithmus übergeben, der zusammen mit dem nächsten Kamerabild in der herkömmlichen Weise die Karte ergänzt und sich innerhalb dieser orientiert.

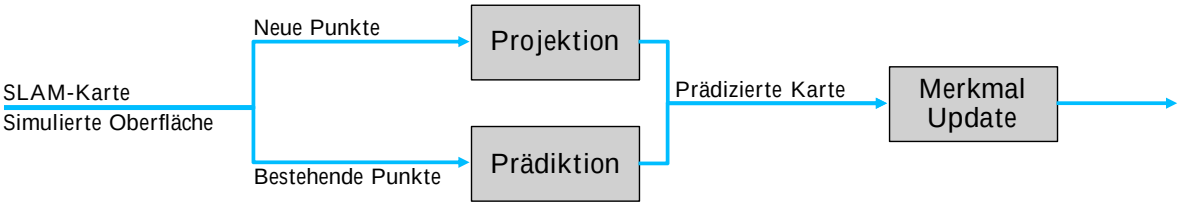

Abbildung 3.8: Vom SLAM-Algorithmus geschätzte, neue Punkte werden unter Berücksichtigung der vom Modell vorhergesagten Verformung auf die Oberfläche des Objekts projiziert, während die Position bestehender Punkte entsprechend angepasst wird. Für alle Punkte, die in der aktuellen Aufnahme zu sehen sind, wird die zugehörige Merkmalposition im Bild durch Rückprojektion in die Bildebene aktualisiert.

Aus dem Blockschaltbild wird zudem ersichtlich, dass das Konzept nur anhand von Simulationsdaten getestet wurde, da nicht nur die Modell Prädiktion, sondern auch die reale Welt durch eine SOFA-Simulation dargestellt wird. Diese Vereinfachung ist für das Testen des erarbeiteten Ansatzes sinnvoll, da somit sichergestellt ist, dass das Modell der Realität entspricht und die Verformung entsprechend vorhergesagt werden kann. In der Umsetzung handelt es sich dabei um zwei getrennte Simulationen, die wie dargestellt miteinander verbunden sind. Das heißt, dass das Framework grundsätzlich auch für reale Kameradaten anwendbar wäre, hierfür allerdings die Genauigkeit des Modells und Kenntnis der auf das Objekt wirkenden Kräfte nicht gegeben sind.

Als Basis des Modell-basierten Ansatzes wurde die MATLAB-Implementierung [Mat21] des klassischen ORB-SLAM nach [MMT15] verwendet. Die Funktionen des gelben Blockes wurden ebenfalls in MATLAB implementiert, sie erhalten so direkt die aktuellen Ergebnisse des Algorithmus. Um den Austausch zwischen SOFA und MATLAB zu realisieren, wird die MATLAB Engine für Python benutzt, so dass die Gesamtsteuerung Python obliegt. Sofa ist zwar in C++ implementiert, kann aber auch über Python gesteuert werden. Es werden also die SOFA-Simulationen initialisiert, gestartet und Bilddaten sowie prädizierte Deformationen generiert. Anschließend werden die MATLAB-Funktionen von Python aufgerufen und die entsprechenden Parameter übergeben. Die Visualisierung der Ergebnisse erfolgt direkt in MATLAB.

Zum Testen wurden verschiedene Kameratrajektorien für die in SOFA enthaltene Kamera vorgegeben. Diese sind im Wesentlichen durch Wegpunkte festgelegt, die aus einer Kamera-

Abbildung 3.9: Erweitertes Blockschaltbild des Modell-basierten SLAM-Algorithmus. Die Ergebnisse des SLAM-Algorithmus werden durch die in Kapitel 3.4 beschriebenen Maßnahmen mit denen der Modell Prädiktion vereint und anschließend an das klassische Verfahren für den nächsten Rechenschritt übergeben. Diese beiden Schritte sind in MATLAB implementiert, das Gesamtframework wird über Python gesteuert.

pose, also Position und Ausrichtung, bestehen und zu einem vorgegebenen Zeitpunkt erreicht werden. Die Wegpunkte sind durch lineare Interpolation verbunden. Durch dieses einfache Vorgehen lässt sich eine Vielzahl an Bewegungen mit unterschiedlicher Geschwindigkeit realisieren. Das untersuchte SOFA-Modell ist eine Art Ellipsoid, das in seiner Grundform der Blase ähnelt (siehe Abbildung 3.3). Da es sich um ein einzelnes Objekt handelt, wurde kein Kollisionsmodell initialisiert - stattdessen kann das Objekt durch frei wirkende Kräfte verformt werden. Wird beispielsweise eine Flächenkraft gewählt, so wirkt diese auf alle im entsprechenden Bereich liegenden Knotenpunkte des Modells und führt so zur Verformung. Darüber hinaus wurden dem Modell beliebige Materialparameter für die Steifigkeit und Elastizität zugewiesen. Wichtig zu betonen ist, dass die Kraft auf das Objekt in beiden Simulationen wirkt - also sowohl der „realen Welt“ als auch der „Modell Prädiktion“. Aufgrund der Tatsache, dass es sich um eine rein synthetische Szene handelt, wurde für die Textur ein einfaches Bild verwendet, das sich durch eine große Vielzahl an Details auszeichnet. Dadurch soll sichergestellt werden, dass die Lokalisierung und Kartierung im starren Fall zufriedenstellend funktioniert und nicht durch fehlerhafte Merkmalerkennung oder -zuordnung negativ beeinträchtigt wird. Nach jedem Simulationsschritt wird ein Bild der Kameraansicht der „realen Welt“ aufgenommen und an den SLAM-Algorithmus übergeben; zudem lässt sich aus der für die Prädiktion zuständigen Simulation die Position der Knotenpunkte auslesen.

3.6 Simulationsergebnisse

Im Folgenden werden beispielhaft zwei Kameratrajektorien mit unterschiedlichem Kraftverlauf genauer untersucht, bevor anhand weiterer Ergebnisse Vor- und Nachteile des gewählten

Ansatzes diskutiert werden. Dabei wird jeweils zunächst der klassische SLAM-Algorithmus im undeformierten System getestet. Anschließend werden der klassische und der Modell-basierte Ansatz bei Verformung angewandt. Zur Bewertung der Performance wird die Wurzel des mittleren quadratischen Fehlers (RMSE, *root mean square error*) der Kameraposition herangezogen. Dazu wird die tatsächliche Kameraposition aus der Simulation bestimmt und in das SLAM-Koordinatensystem transformiert.

3.6.1 Trajektorie 1 - Linearer Kraftverlauf

Im ersten betrachteten Fall wirkt eine linear zunehmende Kraft auf das Objekt und bewirkt dort eine Verformung nach außen, so dass eine Art Beule entsteht. Die Kraft nimmt dabei bis kurz vor Ende der Bewegung stetig zu und bleibt für den letzten Teil konstant. Die entstehende Verformung stellt sich hier also erst nach einiger Zeit ein, zwischen aufeinanderfolgenden Bildern ist nur ein geringer Unterschied zu erkennen. Betrachtet man das Objekt nach absolvierter Trajektorie, dann ist die Ausbeulung dennoch deutlich sichtbar.

In Abbildung 3.10 sind für alle drei Fälle - klassischer Algorithmus ohne Deformation, klassischer Algorithmus mit Deformation und Modell-basierter Ansatz mit Deformation - jeweils die tatsächliche sowie die geschätzte Trajektorie und die resultierende Karte abgebildet. Wie zu erwarten, ist die Genauigkeit der Lokalisierung für den starren Fall sehr hoch und auch die Karte lässt die tatsächliche Form des Ellipsoids gut erkennen. Sobald das System allerdings verformt wird, verschlechtern sich beide Aspekte: Die Lokalisierung erfolgt nur noch mit Abweichung, die mit zunehmender Zeit eher zu- als abnimmt, und die Karte ist äußert verrauscht. In der linken, mittleren Abbildung ist durch das orangene Rechteck der Bereich, in dem die Deformation sichtbar sein sollte, markiert. Hier ist aber lediglich mehr Rauschen zu erkennen und es kann nicht mit Sicherheit gesagt werden, wie die Oberfläche verläuft. Betrachtet man dagegen die Karte des Modell-basierten Ansatzes, so lässt sich besagte Region leicht ausmachen. Noch dazu erweist sich die übrige Karte als rauscharm, da bis auf wenige Ausnahmen alle Punkte auf die Oberfläche des Objekts projiziert wurden. Wie unschwer zu erkennen ist, ist auch die Nachverfolgung der Kamerabewegung deutlich zuverlässiger, was sich auch im durchschnittlichen Fehler niederschlägt, der beim Modell-basierten Konzept bei 0.0179 liegt und damit knapp 82.6 % unter dem Ergebnis des klassischen Ansatzes bleibt (0.103). An die Genauigkeit in starrer Umgebung (0.0125) kommt der Modell-basierte Algorithmus aber nicht ganz heran.

3.6.2 Trajektorie 2 - Sprunghafter Kraftanstieg

In der Operationsumgebung wirken oft nur unmittelbare Kräfte, beispielsweise beim Schneiden des Gewebes. Obwohl sie auf lange Sicht keine allzu große sichtbare Veränderung bewirken, können ihre Auswirkungen für einen kürzeren Zeitraum durchaus beachtlich sein. Um dieses Verhalten darzustellen, wurde für die zweite Trajektorie ein sprunghafter Kraftanstieg nach circa einem Viertel der Bewegung gewählt, wobei sich auch die ausgeführte Kamerabewegung von der vorherigen unterscheidet. Das Resultat ist ein für einige Zeitschritte und damit auch Bilder andauerndes Zucken, das in ähnlicher Form auch in Endoskopieaufnahmen beobachtet werden kann. Im weiteren Verlauf ist die Deformation kaum zu erkennen, dennoch zeigen die Ergebnisse des klassischen Verfahrens in der Mitte von Abbildung 3.11 eine deutliche Abweichung, die sich genau im Moment der Kraftausübung einstellt und nicht

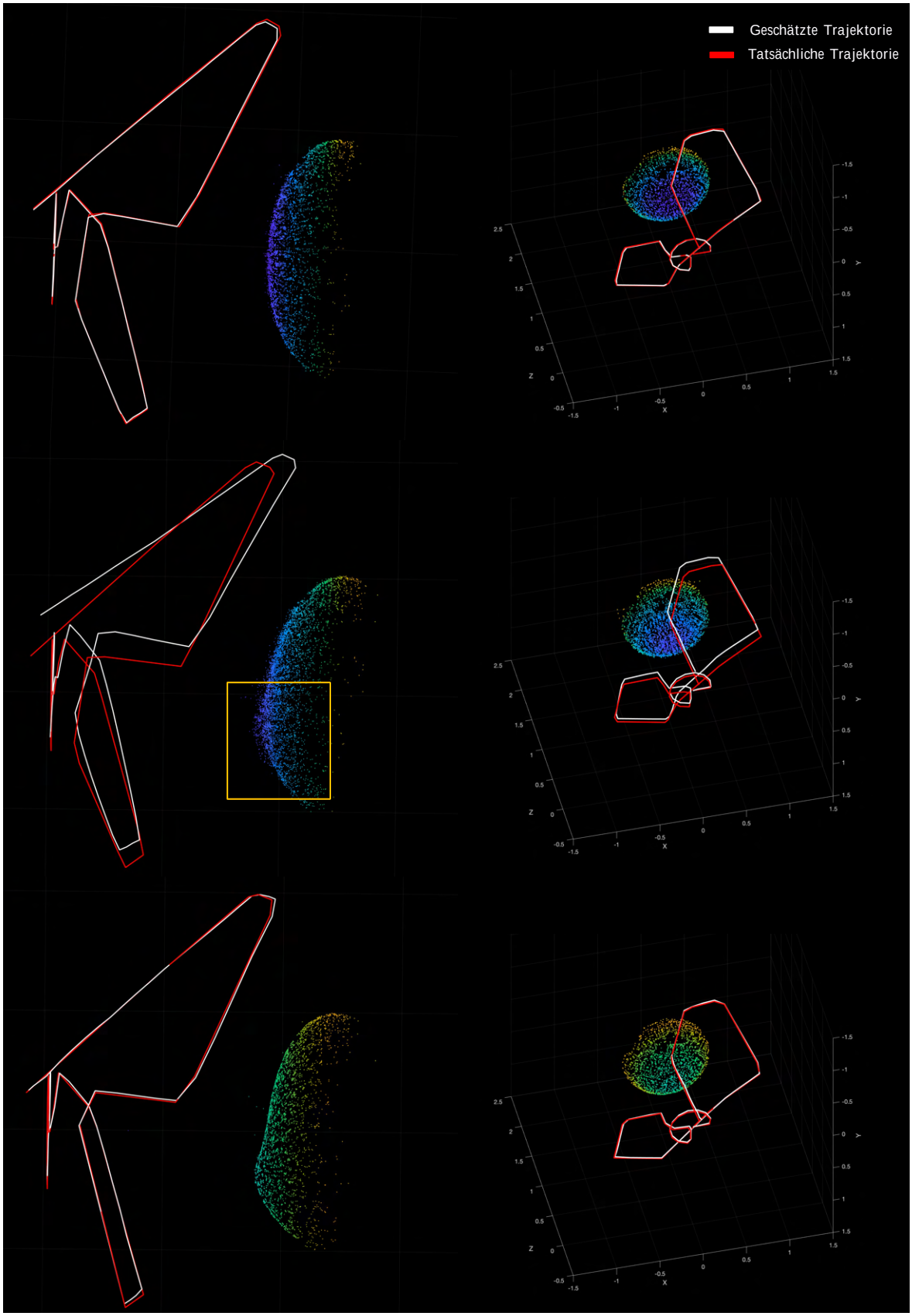

Abbildung 3.10: Ergebnisse des klassischen SLAM-Algorithmus in starrer (Oben) und verformbarer (Mitte) Umgebung und des Modell-basierten Ansatzes bei gleicher Deformation (Unten). Dargestellt ist jeweils die resultierende Karte, wobei die Farbe der erkannten Punkte lediglich zur Visualisierung dienen, und die geschätzte (Weiß) und tatsächliche Kameratrajektorie (Rot).

ausgeglichen werden kann. Der Zeitpunkt, an dem die Kraft auftritt, lässt sich dabei deutlich erkennen: Die Kamera startet zentral vor dem Objekt und nähert sich diesem zunächst relativ gerade an, bevor zwei Kreisbewegungen mit wachsendem Radius ausgeführt werden. Etwa bei der Hälfte dieser Kreisbewegungen kommt es zur Verformung und die geschätzter Position unterscheidet sich erheblich von der tatsächlichen. Der RMSE spiegelt dies auch in Form einer 345-prozentiger Zunahme von 0.0095 auf 0.0423 wider, die sich mit dem klassischen Algorithmus im Vergleich zum starren Fall ergibt. Der Modell-basiert Ansatz schafft es, diesen Fehler fast komplett zu vermeiden, die Wurzel seiner durchschnittlichen quadratischen Abweichung liegt nur bei 0.0109 und damit nur knapp über der im unverformten Fall. Zudem ist die resultierende Karte auch hier deutlich rauschärmer, sie entspricht bis auf wenige Ausreißer der tatsächlichen Oberfläche.

Abbildung 3.11: Ergebnisse für die zweite Trajektorie. Links der klassische SLAM-Algorithmus ohne Deformation, in der Mitte mit Deformation und rechts der Modell-basierte Ansatz mit Deformation. Dabei jeweils in weiß die geschätzte Trajektorie und in rot die tatsächliche Kamerabewegung.

3.6.3 Weitere Simulationsergebnisse und Diskussion

Zusätzlich wurden weitere Szenarien in Form von verschiedenen Kraftverläufen und Kamerabewegungen getestet. Dabei wurde Wert darauf gelegt, unterschiedliche Deformationen durch Variation von Richtung, Betrag und Angriffsfläche der Kräfte zu berücksichtigen. Gleichzeitig ist es von Bedeutung, Fälle darzustellen, bei denen die Verformung unmittelbar in den Aufnahmen ersichtlich ist, aber auch solche, bei denen die Kamera gerade einen Bereich betrachtet, der keiner Veränderung obliegt. Die Ergebnisse zeigen hier einige Variabilität, was an zwei weiteren, beispielhaften Trajektorien erläutert werden soll, deren RMSE im Diagramm in Abbildung 3.12 für Trajektorie 3 und 4 dargestellt ist. Das Diagramm enthält zusätzlich auch die Ergebnisse der beiden bereits erläuterten Trajektorien 1 und 2.

Trajektorie 3 entspricht der Kamerabewegung von Trajektorie 2, allerdings wurde hierfür ein linearer Kraftverlauf gewählt. Wie der Fehler zeigt, bringt der Modell-basierte SLAM-Algorithmus hier eine erhebliche Verschlechterung gegenüber dem starren Fall mit sich. Grund hierfür sind vor allem eine für den Kraftverlauf unvorteilhafte Kamerabewegung, bei der die deformierte Region aus fehleranfälligen Posen betrachtet wird, die vor allem die Projektion erschweren. Zudem zeigen sich hier Schwachstellen beim Finden von Merkmalübereinstimmun-

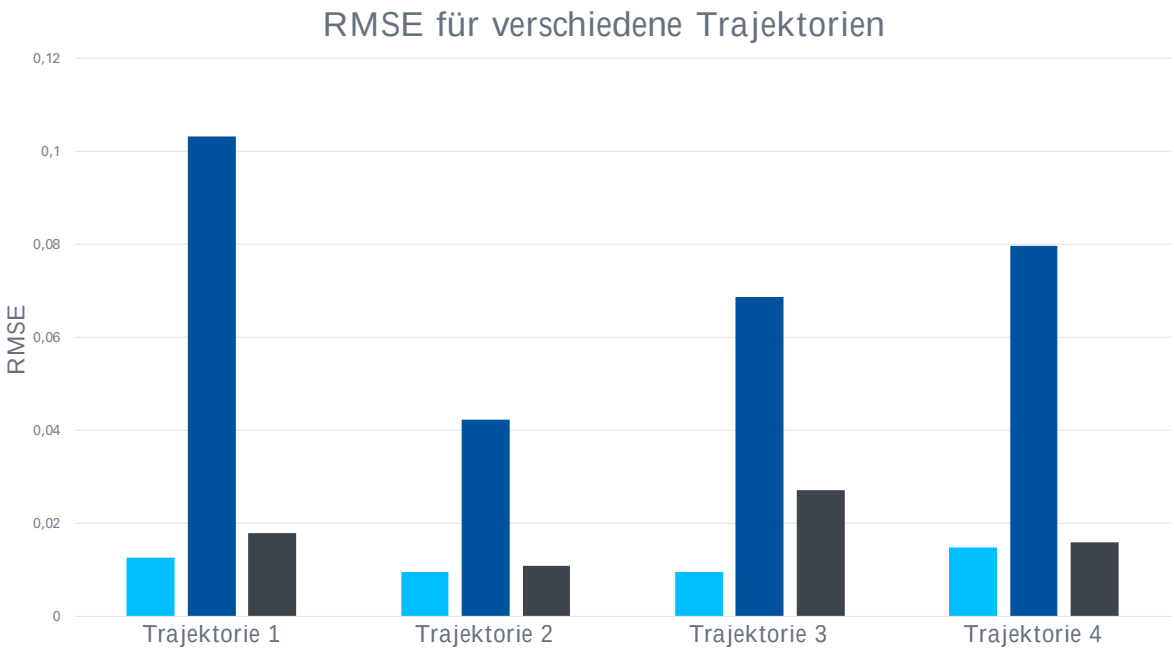

Abbildung 3.12: Vergleich der Wurzel des mittleren quadratischen Fehlers für vier verschiedene Trajektorien. Dargestellt ist jeweils das Resultat des klassischen Algorithmus in starrer (türkis) und verformter (dunkelbau) Umgebung, sowie das Ergebnis des Modell-basierten Ansatzes (dunkelgrau).

gen. Der gewählte, intuitive Ansatz der Rückprojektion auf die Bildebene führt bei größerer Abweichung zwischen zwei Aufnahmen zu Ungenauigkeiten. Problematisch sind dabei zwei Dinge: Zum einen sind nicht für alle in einem Bild erkannten Merkmale zugehörige Raumpunkte verfügbar, deren neue Position durch das Modell vorhergesagt werden kann. Nur für diese Punkte kann aber die zweidimensionale Pixelposition prädiziert werden, während die restlichen Merkmalpositionen zum Beispiel durch Interpolation geschätzt werden müssten. Die zweite Schwierigkeit ergibt sich durch die Funktionsweise des klassischen Algorithmus. Dieser nutzt zur Lokalisierung der Kamera nur die Übereinstimmungen mit dem vorherigen Bild, für die Ergänzung der Karte werden jedoch auch weitere Aufnahmen zu Rate gezogen. Da das aktuelle Verfahren relativ rechenaufwändig ist, werden die Kartenpunkte in der derzeitigen Implementierung lediglich in das letzte Bild projiziert. Die Unterschiede zu früheren Aufnahmen werden hier also genauso wenig kaschiert, wie im klassischen SLAM-Algorithmus. Ein möglicher Lösungsansatz wird in Kapitel 4 kurz vorgestellt.

Anhand dieser Betrachtungen zeigt sich auch, wieso der Modell-basierte Algorithmus mit dem sprunghafte Kraftanstieg in Trajektorie 2 gut zurechtkam. Die Auswirkungen dieser Anregung sind nur für einige Aufnahmen sichtbar. Während dieser Zeitspanne funktioniert die Rückprojektion weitestgehend zuverlässig und ermöglicht so eine fortführende Lokalisierung der Kamera. Nach Abklingen der unmittelbaren Verformung stellt sich hier quasi eine leicht veränderte, starre Umgebung ein, in der der Algorithmus keine Probleme hat. Der klassische SLAM-Algorithmus hat hier aber das Problem, dass seine geschätzte Trajektorie bereits durch das auftretende Zucken von der tatsächlichen abweicht und dieser Fehler anschließend nicht mehr ausgeglichen werden kann.

Ein sehr vielversprechendes Ergebnis stellt Trajektorie 4 dar, bei der der Modell-basierte An-

satz auch für einen linearen Kraftverlauf die nahezu identische Genauigkeit wie das klassische Verfahren in starrer Umgebung aufweist. Hierbei handelt es sich um eine in Hinblick auf die sichtbare Deformation vergleichsweise vorteilhafte Kamerabewegung, bei der die entsprechende Region oft nicht im Sichtfeld der Kamera liegt. Dadurch stellt die Rückprojektion auf die Bildebene hier ein geringeres Problem dar und der Modell-basierte SLAM-Algorithmus liefert die gewünschte Präzision.

Insgesamt lässt sich sagen, dass es zwar einige Kombinationen aus Kameratrajektorie und Kraftverlauf gibt, die für den gewählten Ansatz problematisch sind, die Genauigkeit aber außer in Ausnahmefällen immer deutlich über der des klassischen Verfahrens liegt. Solche Ausnahmefälle ergeben sich vor allem durch sehr deutliche Verformungen, bei denen die Unterschiede zwischen den Bildern maßgeblich sind. Für Trajektorie 3 bringt der Modell-basierte Ansatz immer noch eine Verbesserung in Form einer Reduzierung des RMSE von knapp 60 % gegenüber dem klassischen Verfahren mit sich. Das offensichtlichste Problem stellt in diesen Fällen der gewählte Ansatz der Rückprojektion dar. Allerdings muss festgehalten werden, dass die Grenze für das Versagen des Verfahrens deutlich höher als bei klassischen Ansätzen liegt, die bereits durch leichte Veränderungen in der Umwelt unzureichende Ergebnisse liefern. In diesen Szenarien liefert der Modell-basierte Algorithmus deutlich bessere Ergebnisse.

Zuletzt sei noch darauf hingewiesen, dass die grundsätzliche Performance von kamerabasierten Lokalisierungsalgorithmen immer auch auf der Qualität der Aufnahmen sowie der verwendeten Kalibrierung beruhen. Sind beispielsweise bestimmte Bereiche unzureichend zu erkennen, werden sich diese Ungenauigkeiten auch im Ergebnis niederschlagen. Die Bedeutung der Kalibrierung wurde am Rand von Kapitel 2.2 bereits erwähnt, da sie für die Ausrichtung der Bildebene gegenüber der Kamera von hoher Relevanz ist und während der Berechnung unmittelbaren Einfluss auf die mittels Triangulation bestimmten Raumpunkte nimmt.

Kapitel 4

Zusammenfassung und Ausblick

Diese Arbeit hat sich ausgehend von der kameragestützten Tumorentfernung mit der Frage beschäftigt, wie es möglich ist, sich in verformbaren Umgebungen zu orientieren. Dabei sind zwei Aspekte zentral: Die Lokalisierung der Kamera selbst und die Kartierung der zuvor unbekannten Umgebung. Aus der klassischen Robotik kann hierfür der SLAM-Algorithmus genutzt werden, der für die gleiche Fragestellung entwickelt wurde. Allerdings sind bestehende Ansätze lediglich für starre Umwelten konzipiert, was in der dynamischen Operationsumgebung zu Ungenauigkeiten und letztlich zur Untauglichkeit der klassischen Verfahren führt. Um diese Probleme zu beheben, wurde ein Modell-basierter Ansatz gewählt, der in einer parallelen FEM-Simulation die Deformation des betrachteten Objekts vorhersagt und diese Information in den SLAM-Algorithmus einbezieht. Dafür werden die geschätzten Punkte zunächst auf die Oberfläche des Modells projiziert, so dass die Karte der Oberfläche der Umgebung entspricht. Durch die Verwendung baryzentrischer Koordinaten können auch weitere Verformungen berücksichtigt und die Karte so zu jeder Zeit aktuell gehalten werden. Um jedoch weiterhin die Lokalisierung der Kamera zu ermöglichen, ist es notwendig, die Modell Prädiktion auch in der Bildebene einzubeziehen - schließlich werden hierfür übereinstimmende Merkmale in den Aufnahmen genutzt. Da die Bestimmung der Raumpunkte, aus denen die Karte besteht, mittels Triangulation aus den Bilddaten gewonnen werden, wird hierfür der umgekehrte Weg von Raumpunkt zurück in die Bildebene gewählt. Es erfolgt also eine Rückprojektion ins Bild, was konzeptionell einer zweiten Aufnahme aus gleicher Kameraperspektive nach der Deformation entspricht. Um die gewählten Methoden zu testen, wurde das Gesamtframework des Modell-basierten SLAM-Algorithmus erarbeitet und implementiert. Mittels Simulation wurden der Effekt der gewählten Maßnahmen untersucht. Hierfür wurde, unter Annahme eines exakten Modells und genauer Kenntnis der auf das Objekt wirkenden Kräfte, die Genauigkeit des klassischem Verfahrens jeweils mit und ohne Verformung mit den Ergebnissen des vorgestellten Konzepts verglichen. Dabei zeigt sich, dass einerseits die Karte deutlich rauschärmer und andererseits die Lokalisierung der Kamera genauer ist.

Offensichtlich sind diese Ergebnisse und das zugrundeliegende Verfahren nicht direkt auf realistischere Anwendungsszenarien übertragbar. Dennoch unterstreichen sie die Wirksamkeit des gewählten Ansatzes und des resultierenden Gesamtverfahrens. Ziel der Arbeit war es, dieses Konzept auszuarbeiten, die einzelnen Schritte und das zugehörige Framework zum Datenaustausch zu implementieren und seine grundsätzlichen Vorteile mithilfe der synthetischen Umgebung zu zeigen. Die Arbeit ist daher eher als *Proof of Concept* zu verstehen. Großer Wert wurde auf die Verständlichkeit und leichte Erweiterung des Frameworks gelegt, dass in zukünftigen Arbeiten an einigen Stellen erweitert oder umgestaltet wird. Um für den Operationsfall anwendbar zu sein, gilt es, das Verfahren zunächst in Hinblick auf

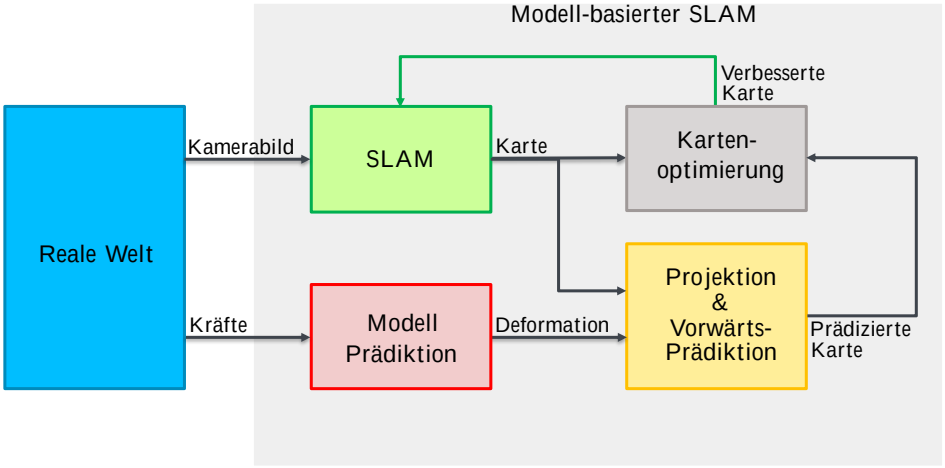

Abbildung 4.1: Blockschaltbild des erweiterten Grundkonzepts des Modell-basierten SLAM-Algorithmus. Da Modell und Realität in der Anwendung nicht übereinstimmen, braucht es eine Entscheidungsinstanz, die zwischen den Ergebnissen des klassischen Verfahrens und der simulativen Prädiktion vermittelt, um zu einer verbesserten Karte und Lokalisierung zu kommen.

seine Anwendung mit realen Kameradaten zu ergänzen. Dafür sind mehrere Schritte notwendig: Obwohl die Simulationsergebnisse vielversprechend sind, zeigen sie doch, dass der Modell-basierte Algorithmus noch einige Schwachstellen hat, die auch in der synthetischen Umgebung zutage treten. Das größte Problem stellt hier die Merkmalzuordnung dar. Der gewählte, intuitive Ansatz der Rückprojektion auf die Bildebene funktioniert in einigen Fällen gut, kann aber bei deutlicher Veränderung zwischen zwei Aufnahmen zu Ungenauigkeiten führen. Eine Schwierigkeit stellt dabei auch seine vergleichsweise ineffiziente Berechnung dar, die in der derzeitigen Implementierung nur die Aktualisierung des vorherigen Bildes erlaubt. Tatsächlich nutzt der Algorithmus zunächst auch nur die letzte Aufnahme, um die Kamerabewegung vorherzusagen. Um zusätzliche Umgebungspunkte zu rekonstruieren, werden aber anschließend auch weitere Aufnahmen verglichen, was derzeit noch zu Ungenauigkeiten in der Karte bzw. unnötigem Rechenaufwand als Folge von fehlerhaften Übereinstimmungen führt. Eine vielversprechende Möglichkeit das Finden von übereinstimmenden Merkmalen robuster zu gestalten, bieten Graphen-basierte Ansätze, bei denen nicht die einzelnen Merkmale sondern viel mehr die geometrische Struktur miteinander verglichen wird.

Um die Erweiterung des Modell-basierten Ansatzes auf reale Kamerabilder zu realisieren, ist es zudem notwendig, ein tatsächliches Modell der Umgebung zu verwenden, was wiederum dazu führt, dass die Ergebnisse der Projektion und Vorwärts-Prädiktion nicht immer übernommen werden können. Grund hierfür ist, dass, anders als in den getesteten Szenarien, das Modell und daher auch die vorhergesagte Verformung nicht genau der Realität entsprechen werden. Es braucht also eine Instanz, die entscheidet, ob mehr auf das Kamerabild und die daraus abstrahierte Information oder die aus dem Modell gewonnenen Vorhersagen vertraut werden kann. Die Entscheidung kann beispielsweise durch einen Kalman-Filter realisiert werden. Dabei wird voraussichtlich gerade für kleine Deformationen das Modell kaum zusätzlichen Nutzen bieten, da seine Ungenauigkeit hier noch zu groß ist. Für vergleichsweise große Verformungen wird das Modell aber hilfreiche Informationen liefern und so zur Eindämmung des Fehlers des SLAM-Algorithmus beitragen. In diesem Zusammenhang präsentieren auch

die auf das Objekt wirkenden Kräfte eine große Hürde, da diese im Regelfall nicht gemessen werden können, sondern durch einen Beobachter aus den vorliegenden Daten geschätzt werden müssen.

Im Allgemeinen muss die Recheneffizienz des gesamten Verfahrens noch erhöht werden, z.B. durch Multithreading oder effizientere Auswahl der Dreiecke für die Projektion. In Hinblick auf eine mögliche intraoperative Anwendung ist dies notwendig, da die ermittelten Informationen dem Operateur in Echtzeit zur Verfügung gestellt werden müssen, um eine präzise Navigation zu ermöglichen.

Anhang A

Anhang

Um die Abbildung der Umgebungspunkte auf die Bildebene darzustellen, wird in der projektiven Geometrie häufig auf das Lochkameramodell zurückgegriffen, das in Abbildung A.1 dargestellt ist. Dabei werden die Umgebungspunkte über den Brennpunkt auf die Bildebene projiziert, auf der ein Abbild der Umwelt entsteht. Zur einfacheren Betrachtung kann die Bildebene auch ungespiegelt vor dem Brennpunkt dargestellt werden. Neben den extrinsischen Parametern, die in Kapitel 2.2 genauer untersucht werden, spielt vor allem die intrinsischen Kameramatrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.1}$$

eine wichtige Rolle. Sie bestimmt die Umrechnung von 3D-Koordinaten im Kamera-Koordinatensystem zu Pixelkoordinaten im Bild. Die entscheidenden Parameter sind der Bildhauptpunkt (*principal point*) $c = \begin{bmatrix} c_x & c_y \end{bmatrix}^\top$ und die Brennweite (*focal length*) $f = \begin{bmatrix} f_x & f_y \end{bmatrix}^\top$. Der Bildhauptpunkt gibt dabei den Schnittpunkt der Normalen der Brennebene ausgehend vom Brennpunkt mit der Bildebene in Pixelkoordinaten an. Im Regelfall ist dieser mittig auf der Bildebene, c_x entspricht also der halben Bildbreite in Pixeln, c_y der halben Bildhöhe. Die Brennweite beinhaltet Informationen zur Umrechnung von Pixel- zu Welteinheiten, beispielsweise Millimeter. Mittels Kalibrierung werden für die verwendete Kamera diese intrinsischen Parameter vorab bestimmt - sie verändern sich im Gegensatz zu den extrinsischen Parametern nicht, sondern müssen lediglich neu bestimmt werden, wenn eine andere Kamera genutzt wird. Die Genauigkeit der Kalibrierung kann das Ergebnis des SLAM-Algorithmus entscheidend beeinflussen, da die bestimmten Werte für die Skalierung der Rekonstruktion und die Umrechnung von Pixelkoordinaten in der Bildebene auf Umgebungskoordinaten entscheidend sind. Fehlerhafte Kalibrierungen können daher zu einer Verschiebung oder Verzerrung der rekonstruierten Szene führen, deshalb sollte dieser Schritt nicht vernachlässigt werden.

Abbildung A.1: Projektion eines Umgebungspunktes auf die Bildebene mit dem Lochkameramodell, angelehnt an [Sch05].

Abbildungsverzeichnis

Literatur

[BD06] T. Bailey und H. Durrant-Whyte. „Simultaneous localization and mapping (SLAM): part II“. In: *IEEE Robotics Automation Magazine* 13.3 (2006), S. 108–117. DOI: 10.1109/MRA.2006.1678144.

[CB12] Toby Collins und Adrien Bartoli. „Towards live monocular 3D laparoscopy using shading and specularity information“. In: *International Conference on Information Processing in Computer-Assisted Interventions*. Springer. 2012, S. 11–21.

[CL85] R. Chatila und J. Laumond. „Position referencing and consistent world modeling for mobile robots“. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Bd. 2. 1985, S. 138–145. DOI: 10.1109/ROBOT.1985.1087373.

[Cad+16] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid und John J. Leonard. „Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age“. In: *IEEE Transactions on Robotics* 32.6 (2016), S. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[Cal+10] Michael Calonder, Vincent Lepetit, Christoph Strecha und Pascal Fua. „Brief: Binary robust independent elementary features“. In: *European conference on computer vision*. Springer. 2010, S. 778–792.

[DB06] H. Durrant-Whyte und T. Bailey. „Simultaneous localization and mapping: part I“. In: *IEEE Robotics Automation Magazine* 13.2 (2006), S. 99–110. DOI: 10.1109/MRA.2006.1638022.

[Dav03] Andrew Davison. „Real-time simultaneous localisation and mapping with a single camera“. In: Bd. 2. Jan. 2003, S. 1403–1410. DOI: 10.1109/ICCV.2003.1238654.

[EKC17] Jakob Engel, Vladlen Koltun und Daniel Cremers. „Direct sparse odometry“. In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), S. 611–625.

[Eri05] Christer Ericson. „Chapter 5 - Basic Primitive Tests“. In: *Real-Time Collision Detection*. Hrsg. von Christer Ericson. The Morgan Kaufmann Series in Interactive 3D Technology. San Francisco: Morgan Kaufmann, 2005, S. 125–233. ISBN: 978-1-55860-732-3. DOI: https://doi.org/10.1016/B978-1-55860-732-3.50010-3. URL: https://www.sciencedirect.com/science/article/pii/B9781558607323500103.

[FPRARM15] Jorge Fuentes-Pacheco, José Ruiz-Ascencio und Juan Manuel Rendón-Mancha. „Visual simultaneous localization and mapping: a survey“. In: *Artificial intelligence review* 43.1 (2015), S. 55–81.

[Fau+12] Francois Faure, Christian Duriez, Hervé Delingette, Jeremie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik und Stéphane Cotin. „SOFA: A Multi-Model Framework for Interactive Physical Simulation“. In: Bd. 11. Juni 2012. ISBN: 978-3-642-29013-8. DOI: 10.1007/8415_2012_125.

[GVL13] Gene H Golub und Charles F Van Loan. *Matrix computations*. Bd. 3. JHU press, 2013.

[HS97] Richard I Hartley und Peter Sturm. „Triangulation“. In: *Computer vision and image understanding* 68.2 (1997), S. 146–157.

[Hao+15] Nazim Haouchine, Jeremie Dequidt, Marie-Odile Berger und Stephane Cotin. „Monocular 3d reconstruction and augmentation of elastic surfaces with self-occlusion handling“. In: *IEEE transactions on visualization and computer graphics* 21.12 (2015), S. 1363–1376.

[KM07] Georg Klein und David Murray. „Parallel tracking and mapping for small AR workspaces“. In: *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, S. 225–234.

[LM18] Jose Lamarca und Jose Maria Martinez Montiel. „Camera tracking for slam in deformable maps“. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, S. 0–0.

[Lam+21] J. Lamarca, S. Parashar, A. Bartoli und J. M. M. Montiel. „DefSLAM: Tracking and Mapping of Deforming Scenes From Monocular Sequences“. In: *IEEE Transactions on Robotics* 37.1 (2021), S. 291–303. DOI: 10.1109/TRO.2020.3020739.

[Lin+13] Bingxiong Lin, Adrian Johnson, Xiaoning Qian, Jaime Sanchez und Yu Sun. „Simultaneous tracking, 3D reconstruction and deforming point detection for stereoscope guided surgery“. In: *Augmented Reality Environments for Medical Imaging and Computer-Assisted Interventions*. Springer, 2013, S. 35–44.

[Lin+16] Bingxiong Lin, Yu Sun, Xiaoning Qian, Dmitry Goldgof, Richard Gitlin und Yuncheng You. „Video-based 3D reconstruction, laparoscope localization and deformation recovery for abdominal minimally invasive surgery: a survey“. In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 12.2 (2016), S. 158–178.

[MAT17] Raul Mur-Artal und Juan D Tardós. „Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras“. In: *IEEE Transactions on Robotics* 33.5 (2017), S. 1255–1262.

[MBC11] Abed Malti, Adrien Bartoli und Toby Collins. „Template-based conformal shape-from-motion from registered laparoscopic images.“ In: *MIUA*. Bd. 1. 2. 2011, S. 6.

[MMT15] R. Mur-Artal, J. M. M. Montiel und J. D. Tardós. „ORB-SLAM: A Versatile and Accurate Monocular SLAM System“. In: *IEEE Transactions on Robotics* 31.5 (2015), S. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[Mah+16] Nader Mahmoud, Iñigo Cirauqui, Alexandre Hostettler, Christophe Doignon, Luc Soler, Jacques Marescaux und JMM Montiel. „ORBSLAM-based endoscope tracking and 3D reconstruction“. In: *International workshop on computer-assisted and robotic endoscopy*. Springer. 2016, S. 72–83.

[Mah+18] Nader Mahmoud, Toby Collins, Alexandre Hostettler, Luc Soler, Christophe Doignon und Jose Maria Martinez Montiel. „Live tracking and dense reconstruction for handheld monocular endoscopy“. In: *IEEE transactions on medical imaging* 38.1 (2018), S. 79–89.

[Mat21] Mathworks. *Monocular Visual Simultaneous Localization and Mapping (vSLAM) in MATLAB*. `https://de.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html`. Accessed: 2021-04-16. 2021.

[PPB17] Shaifali Parashar, Daniel Pizarro und Adrien Bartoli. „Isometric non-rigid shape-from-motion with riemannian geometry solved in linear time“. In: *IEEE transactions on pattern analysis and machine intelligence* 40.10 (2017), S. 2442–2454.

[Pol04] Marc Pollefeys. „Visual 3D Modeling from Images.“ In: Jan. 2004, S. 3.

[QR18] Liang Qiu und Hongliang Ren. „Endoscope navigation and 3D reconstruction of oral cavity by visual SLAM with mitigated data scarcity“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, S. 2197–2204.

[RD06] Edward Rosten und Tom Drummond. „Machine Learning for High-Speed Corner Detection“. In: *Computer Vision – ECCV 2006*. Hrsg. von Aleš Leonardis, Horst Bischof und Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 430–443. ISBN: 978-3-540-33833-8.

[Rub+11] E. Rublee, V. Rabaud, K. Konolige und G. Bradski. „ORB: An efficient alternative to SIFT or SURF“. In: *2011 International Conference on Computer Vision*. 2011, S. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[SSC90] Randall Smith, Matthew Self und Peter Cheeseman. „Estimating Uncertain Spatial Relationships in Robotics“. In: *Autonomous Robot Vehicles*. Hrsg. von Ingemar J. Cox und Gordon T. Wilfong. New York, NY: Springer New York, 1990, S. 167–193. ISBN: 978-1-4613-8997-2. DOI: 10.1007/978-1-4613-8997-2_14. URL: https://doi.org/10.1007/978-1-4613-8997-2_14.

[Sch05] O. Schreer. *Stereoanalyse und Bildsynthese*. Springer Berlin Heidelberg, 2005. ISBN: 9783540274735. URL: `https://books.google.de/books?id=mIQfBAAAQBAJ`.

[Son+18] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang und Gamini Dissanayake. „Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing“. In: *IEEE Robotics and Automation Letters* 3.4 (2018), S. 4068–4075.

[Thr08] Sebastian Thrun. „Simultaneous Localization and Mapping“. In: *Robotics and Cognitive Approaches to Spatial Mapping*. Hrsg. von Margaret E. Jefferies und Wai-Kiang Yeap. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 13–41. ISBN: 978-3-540-75388-9. DOI: 10.1007/978-3-540-75388-9_3. URL: https://doi.org/10.1007/978-3-540-75388-9_3.

[Tur+17] Mehmet Turan, Yasin Almalioglu, Helder Araujo, Ender Konukoglu und Metin Sitti. „A non-rigid map fusion-based direct SLAM method for endoscopic capsule robots“. In: *International journal of intelligent robotics and applications* 1.4 (2017), S. 399–409.

[VSSY12] Marco Visentini-Scarzanella, Danail Stoyanov und Guang-Zhong Yang. „Metric depth recovery from monocular images using shape-from-shading and specularities“. In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, S. 25–28.

[Vel+16] Vimalraj Velayutham, David Fuks, Takeo Nomi, Yoshikuni Kawaguchi und Brice Gayet. „3D visualization reduces operating time when compared to high-definition 2D in laparoscopic liver resection: a case-matched study“. In: *Surgical endoscopy* 30.1 (2016), S. 147–153.

[WNJ10] Chenyu Wu, Srinivasa G Narasimhan und Branislav Jaramaz. „A multi-image shape-from-shading framework for near-lighting perspective endoscopes“. In: *International Journal of Computer Vision* 86.2 (2010), S. 211–228.

[WT00] Wei Wang und Hung Tat Tsui. „A SVD decomposition of essential matrix with eight solutions for the relative positions of two perspective cameras“. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Bd. 1. IEEE. 2000, S. 362–365.

[Wag+12] OJ Wagner, Monika Hagen, A Kurmann, S Horgan, Daniel Candinas und SA Vorburger. „Three-dimensional vision enhances task performance independently of the surgical method“. In: *Surgical endoscopy* 26.10 (2012), S. 2961–2968.

[Wei+13] Wei Tan, Haomin Liu, Z. Dong, G. Zhang und H. Bao. „Robust monocular SLAM in dynamic environments“. In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2013, S. 209–218. DOI: 10.1109/ISMAR.2013.6671781.

[XZ13] Gang Xu und Zhengyou Zhang. *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Bd. 6. Springer Science & Business Media, 2013.

[Zha98] Zhengyou Zhang. „Determining the epipolar geometry and its uncertainty: A review“. In: *International journal of computer vision* 27.2 (1998), S. 161–195.

Erklärung des Autors

der Bachelorarbeit mit dem Titel

A Model-based Simultaneous Localization and Mapping Approach for Deformable Bodies

Hiermit versichere ich,

1. dass ich meine Arbeit bzw. bei einer Gruppenarbeit den entsprechend gekennzeichneten Anteil der Arbeit selbständig verfasst habe,
2. dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe,
3. dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist,
4. dass ich die Arbeit weder vollständig noch in Teilen bereits veröffentlicht habe und
5. dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

Stuttgart, den 29.03.2021

Jonathan Haag